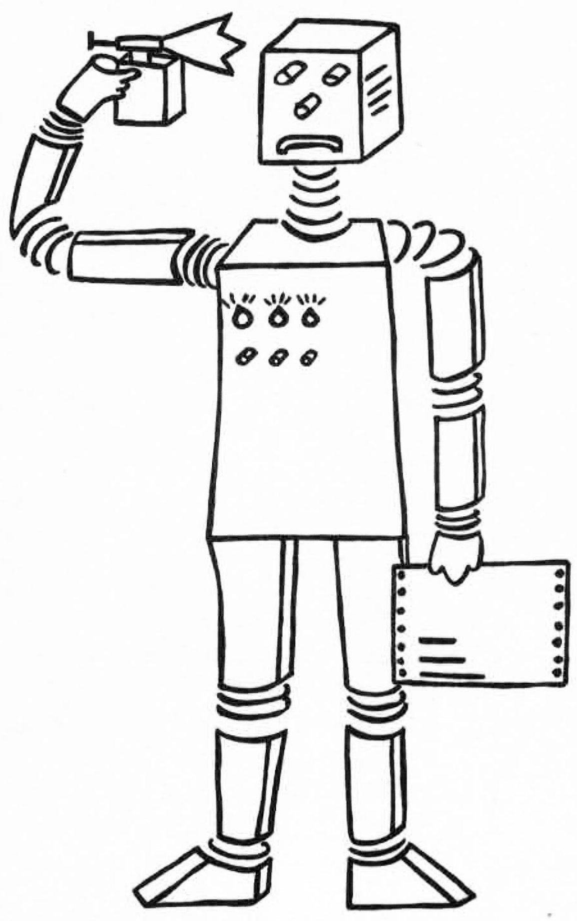
SEKI-REPORT

Artificial
Intelligence
Laboratories

Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern 1, W. Germany



On Solving Equations and Disequations

W. Buntine & H.-J. Bürckert

SEKI Report SR-89-03

On Solving Equations and Disequations

WRAY L. BUNTINE

Kez Centre for Computing Sciences, University of Technology, Sydney,P.O. Box 123, Broadway, 2007 Australia

HANS-JÜRGEN BÜRCKERT

FB Informatik, Universität Kaiserslautern, Postfach 3049, D-6750 Kaiserslautern, FR Germany

Abstract: We are interested in the problem of solving a system $\langle s_i = t_i: 1 \leq i \leq n, p_j \neq q_j: 1 \leq j \leq m \rangle$ of equations and disequations, also known as disunification. Solutions to disunification problems are substitutions for the variables of the problem that make the two terms of each equation equal, but leave those of the disequations different. We investigate this in both algebraic and logical contexts where equality is defined by an equational theory and more generally by a definite clause equality theory E. We show how E-disunification can be reduced to E-unification, that is solving equations only, and give a disunification algorithm for theories given a unification algorithm. In fact this result shows that for theories where the solutions of all unification problems can be represented by finitely many substitutions, the solutions of all disunification problems can also be represented finitely. We apply disunification to handle negation in logic programming with equality in a similar style to Colmerauer's logic programming with rational trees, and to represent many solutions to AC-unification problems by a few solutions to ACI-disunification problems.

Keywords: Equational theory, definite clause, logic programming, E-unification, E-disunification, solving equations and disequations, inequations.

1. Introduction

The problem of solving equations in a given equational theory, also known as E-unification (Plotkin 1972, Siekmann 1978, 1989, Huet & Oppen 1980, Fages & Huet 1983, 1986, Bürckert et al. 1989) has many applications in artificial intelligence and computer science (see Siekmann 1989 for an overview). Closely related is the problem of solving disequations (negated equations), that is, finding solutions for problems of the form $\langle p \neq q \rangle$, or more generally solving a combination of equations and disequations

$$\langle s_i = t_i: 1 \leq i \leq n, p_j \neq q_j: 1 \leq j \leq m \rangle.$$

Solutions to these problems are substitutions of the variables in the terms s_i, t_i, p_j, q_j, such that the instantiated terms of the equations become equal, while the instantiated terms of the disequations remain different – both with respect to a given equational theory. In the special case of the empty (equational) theory equality and difference are defined purely syntactically (Robinson 1969, Colmerauer 1984).

Originally A. Colmerauer (1984) discussed the problem of solving equations *and* disequations in the framework of logic programming, and gave an algorithm to solve these disunification problems. H. Comon (1986) investigated the problem for certain applications in algebraic specification. In fact he proposed it as a tool for proving sufficient completeness of an algebraic specification defined by a set of rewrite rules. C. Kirchner & P. Lescanne (1987) and J.-L. Lassez, M.J. Maher and K. Marriot (1987) again took up the problem in a more general framework of a family of alternative systems of equations and disequations, where certain variables play the role of parameters, that is, solutions must satisfy at least one of the alternative systems for all instantiations of the parameter variables. All these papers, however, deal with uninterpreted signatures, that is, they investigate solving equations and disequations with respect to syntactical equality of terms.

We want to generalize disunification for arbitrary equational theories, and more generally for definite clause equality theories. This is necessary to represent awfully many possible solutions to a general unification problem by a far smaller number of solutions to a suitable corresponding disunification problem. Two examples will demonstrate how this may work.

Our first example is in the theory of logic programming. Here under current investigation is the augmentation of standard logic programming languages with equality, and with negation. This can be done by replacing standard unification with E-unification (Goguen & Meseguer 1984, Bürckert 1986, Gallier & Raatz 1986, Jaffar & Stuckey 1986), and by the concept of completed logic programs allowing a restricted form of negation, negation as failure (Clark 1978). For a combination of both augmentations the equality theory has to be completed as well. This is based on the concept of equality completion or unification complete equality theories (Jaffar et al. 1984). Further extensions, however, are required to the theory of equality, unification and the like to advance this treatment further. For instance, solutions to negative questions may really require negated equations in their expression: A full solution to the question "which x are *not* member of the list $(1, 2)$?" is the negative answer "$x \neq 1 \wedge x \neq 2$", instead of the common approach of generating all infinitely many possible positive answers – with respect to some fixed domain of discourse (here for example the infinitely many positive integers "$x = 3$ or $x = 4$ or ...", when the membership relation is defined for lists of positive integers only). Thus we can represent awfully many answer substitutions by a few, however, more general solutions – in our approach substitutions together with simplified disequations (negated substitutions) as constraints (cf. A. Colmerauer's "simplified systems").

The second example concerns part of unification theory itself, the task of designing efficient unification algorithms. There are many implementations of unification algorithms for a theory of associative and commutative function symbols (AC-unification), the very recent are highly efficient. But already small AC-unification problems may have thousands of AC-unifiers. For instance, the AC-unification problem

$$\langle x{\cdot}y{\cdot}z = v{\cdot}v{\cdot}v{\cdot}v \rangle_{AC}$$

has *32 000* most general AC-unifiers ("$\cdot$" is a binary, associative, and commutative function symbol written infix, parentheses are dropped), and a similar one

$$\langle x{\cdot}y{\cdot}z{\cdot}v = w{\cdot}w{\cdot}w \rangle_{AC}$$

has more than *a million* most general solutions (Bürckert et al. 1989₂). The generation of all these AC-unifiers can be avoided by representing them as solutions to certain AC1-disunification problems (associative, commutative functions with a unit). For example, the AC1-disunification problem

$$\langle x{\cdot}y{\cdot}z = v{\cdot}v{\cdot}v{\cdot}v\ ,\ x \neq 1,\ y \neq 1,\ z \neq 1,\ v \neq 1\ \rangle_{AC1}$$

has exactly the same substitutions as solutions as the first of the above AC-unification problems; for example, the substitution

$$\{x \leftarrow w,\ y \leftarrow w,\ z \leftarrow v{\cdot}v\}$$

is such a common solution, since it solves the equation with respect to both theories *AC* and *AC1* and it substitutes none of the variables by the unit *1*. This correspondence occurs because the AC-solutions of an equation are exactly those instances of the AC1-solutions of the same equation that do not instantiate any of the unknowns by the unit (Livesey & Siekmann 1975). However, the equation in our example problems have only *one* most general AC1-unifier (notice, that this is the case for all problems containing only variables). The idea is again to represent the many solutions of these AC-unification problems or the equivalent AC1-disunification problems by the few (possibly one) AC1-unifiers, adding the disequations as constraints to them (section 5.3).

Recently some further results dealing with solving disequations over equational theories came up (Comon & Lescanne 1987, Comon 1988). They investigate parameterized unification problems consisting of disjunctions of systems of equations and disequations, where some of the variables are not treated as unknowns but as parameters. H. Comon and P. Lescanne give a set of rules defining a disunification algorithm for the above kind of problems in the case of the syntactical theory and the theory *AC* of associativity and commutativity. H. Comon has a complete disunification procedure for any theory with an existing unification algorithm, and a disunification algorithm for certain special theories he calls "quasi-free". He also has a disunification algorithm for parameterless problems in theories that are finitary with respect to unification. In both papers solved forms are used that differ from ours depending on different applications.

In the following sections we give a formal framework for solving equation and disequations in an arbitrary equational or definite clause equality theory *E*. We introduce the necessary algebraic and model theoretic notations (section 2.1) in order to define E-disunification problems and their solutions, and we recapitulate the notations of unification theory in equational theories (section 2.2). We consider solving equations and disequations from an algebraic viewpoint, i.e., solving them in a special fixed algebra (section 3.1), generalize unification theory to more general equality theories (section 3.2), and then discuss disunification with respect to a more general closed world assumption, the equality completion approach (section 3.3). Since there is some trouble with the notion of solutions defined by substitutions in the case of disequations, we generalize the notion of solutions to solved forms (section 4.1), that are pairs of a substitution together with a family of exceptions or constraints for this substitution. These exceptions and constraints are also defined in terms of ("negated") substitutions, and can be interpreted as simplified forms of disequations, similar to the interpretation of substitutions as simplified forms of equations. We consider the problems of inconsistent generalized "solutions", translation between the two forms of solutions, redundancy, and complete representation of generalized solutions. In section 4.2 we give a representation theorem that shows that we can get the solutions of a disunification problem as the instances of a set of substitutions with exceptions, the exceptions are built up as the solutions of related unification problems (and accordingly for constrained substitutions). In particular, we show that when the solutions of all E-unification problems can be represented by finite sets of substitutions we also have a finite representation for the solutions of all

E-disunification problems by our more general solved forms. This leads to an algorithm for solving E-disunification problems by computing the solutions for corresponding E-unification problems provided there is a terminating E-unification algorithm. We discuss propagation and merging of our generalized solved forms (section 4.3) and the non-trivial problem of restricting generalized solutions to subsets of the variables (section 4.4). Finally in section 5 we sketch several applications for disunification.

2. Preliminaries

We assume the reader to be familiar with the notions of first order logic, equational logic, and universal algebra as needed for unification theory and logic programming; we are consistent with the notations of the common literature (Shoenfield 1967, Burris & Sankappanavar 1979, Grätzer 1979, Taylor 1979, Huet & Oppen 1980, Goguen & Meseguer 1984, Lloyd 1984, Fages & Huet 1986, Gallier 1986, Kirchner 1989, Siekmann 1989). We just recall the most important notations.

2.1 Algebras, Structures, Theories

A signature Σ is the (disjoint) union of a set $\mathcal{F}$ of *function* symbols and a set $\mathcal{P}$ of *predicate* symbols, where a nonnegative integer (the *arity*) is assigned to each symbol in Σ. Function symbols with arity 0 are also called *constants*. We require the signatures to contain at least one constant and a distinguished binary predicate symbol = written infix, the *equality* symbol.

A *Σ-algebra $\mathcal{A}$* consists of a *carrier A* together with *operations $f^{\mathcal{A}}$: $A^n \to A$* for every n-ary function symbol $f \in \Sigma$ – the constants just correspond to distinguished elements in A. A *Σ-homomorphism* is a mapping φ from a Σ-algebra $\mathcal{A}$ to a Σ-algebra $\mathcal{B}$, such that $\varphi f^{\mathcal{A}}(a_1,\dots,a_n) = f^{\mathcal{B}}(\varphi a_1,\dots,\varphi a_n)$. A *$\Sigma$-congruence* is an equivalence relation $\approx$ on the carrier A of a Σ-algebra $\mathcal{A}$, such that $f^{\mathcal{A}}(a_1,\dots,a_n) \approx f^{\mathcal{A}}(b_1,\dots,b_n)$, whenever $a_i \approx b_i$ $(1 \leq i \leq n)$. The *quotient $\mathcal{A}/_{\approx}$* of a Σ-algebra $\mathcal{A}$ by such a Σ-congruence $\approx$ is the Σ-algebra, whose carrier is the set of all congruence classes $a/_{\approx}$ of elements $a \in A$, and whose operations are defined by $f^{\mathcal{A}/\approx}(a_1/_{\approx},\dots,a_n/_{\approx}) := (f^{\mathcal{A}}(a_1,\dots,a_n))/_{\approx}$.

A *Σ-structure* (with equality) is a Σ-algebra, where in addition a *relation $p^{\mathcal{A}} \subseteq A^n$* is assigned to every n-ary predicate symbol $p \in \Sigma$ ($=^{\mathcal{A}}$ is the equality relation in A, i.e., the set $\{(a, a)\colon a \in A\}$); we write $p^{\mathcal{A}}(a_1,\dots,a_n)$, when $(a_1,\dots,a_n) \in p^{\mathcal{A}}$. Notice, that an algebra can also be viewed as a structure over the signature containing only the function symbols and the binary equality symbol, where the last one is assigned with the equality relation on the carrier.

Given a possibly infinite set $\mathcal{V}$ of *variable* symbols the *Σ-terms, Σ-atoms, Σ-formulae over $\mathcal{V}$* are defined as usual (we use the connectors $\wedge$, $\vee$, $\neg$, $\Rightarrow$ (or $\Leftarrow$), $\Leftrightarrow$, and the quantifiers $\forall,\exists$). We assume the free variables of an open formula as universally quantified, if necessary we close an open formula F by the universal or existential quantifier: $\forall.F$ or $\exists.F$. We also use the notations $\forall_X F$ and $\forall_{-X} F$, where X is a set of variables, in order to universally quantify all free variables of F that are in X and that are not in X, respectively; and analogously for the existential quantifier. Sets of formulae are considered as conjunctions of their elements. A *definite clause* is a formula of the form

$H \Leftarrow B_1 \wedge \ldots \wedge B_n$ where H and the B_i are atoms. In this paper a *definite clause theory*, also called *logic program*, is a set of definite clauses. In a *definite clause equality theory*, the only predicate symbol occurring is the equality symbol "=".

The *Σ-term algebra over $\mathcal{V}$* is denoted $\mathcal{T}_\Sigma(\mathcal{V})$ or for short $\mathcal{T}$, when $\mathcal{V}$ is countably infinite and fixed. Its carrier is the set of Σ-terms, and its operations $f^{\mathcal{T}}$ map terms $t_1,\ldots,t_n$ to the term $f(t_1,\ldots,t_n)$. If $\mathcal{V}$ is the empty set, $\mathcal{T}_\Sigma(\emptyset)$ is called the *ground Σ-term algebra*, abbreviated by $\mathcal{T}_0$. The set of Σ-endomorphisms on the term algebra $\mathcal{T}_\Sigma(\mathcal{V})$ that move at most a finite set of variables is denoted by $\mathcal{SUB}_\Sigma$; its elements are called *Σ-substitutions*; ε is the identity on $\mathcal{T}$ or *empty substitution*. We call a Σ-atom $s = t$ a *Σ-equation* and its negation $s \neq t$ a *Σ-disequation*.

For any syntactical object O like atoms, terms, substitutions, etc. we write $Var(O)$ for the set of variables occurring in this object. For any substitution σ the finite set of variables $DOM\sigma := \{x \in \mathcal{V}\colon \sigma x \neq x\}$ is called the *domain*, the finite set of terms $COD\sigma := \{\sigma x\colon x \in DOM\sigma\}$ is the *codomain*, and $VCOD\sigma := Var(COD\sigma)$ is the set of variables *introduced by σ*. We represent a substitution σ by the finite set of its *substitution components* $\{x \leftarrow \sigma x\colon x \in DOM\sigma\}$. Given two subsets V, W of $\mathcal{V}$ we denote the set of substitutions with domain V and introduced variables in W by $\mathcal{SUB}(V,W)$; they are homomorphisms from $\mathcal{T}(V)$ to $\mathcal{T}(W)$. If W is empty, we call the elements of $\mathcal{SUB}(V, \emptyset)$ ground substitutions. A *renaming* of a set of variables V to a set of variables W is a substitution $\rho \in \mathcal{SUB}(V, W)$ that is an injective homomorphism of $\mathcal{T}(V)$ into $\mathcal{T}(W)$; hence it satisfies $DOM\rho = V$, $COD\rho \subseteq W$, and $\rho x = \rho y$ iff $x = y$ for all variables $x,y \in DOM\rho$; its *converse* is the renaming $\rho^c \in \mathcal{SUB}(COD\rho, V)$. The *restriction* $\sigma/_V$ of a substitution σ to a set of variables V is defined by $\sigma/_V x = \sigma x$ for all $x \in V$ and $\sigma/_V x = x$, otherwise.

For every substitution σ there is a corresponding open formula $[\sigma]$ denoting the conjunction $\bigwedge_{x \in Dom\sigma} x = \sigma x$; we also extend this to sets Θ of substitutions, such that $[\Theta]$ denotes the disjunction $\bigvee_{\theta \in \Theta} [\theta]$. Note, that by our convention on open formulae, the above formulae are universally quantified. For an *idempotent* substitution σ – i.e. $\sigma = \sigma\sigma$ or equivalently $DOM\sigma \cap VCOD\sigma = \emptyset$ – the formulae $\exists_X\, [\sigma]$ and $\exists_X\, [\sigma|_{DOM\sigma - X}]$ as well as $\forall.\sigma F$ and $\forall.[\sigma] \Rightarrow F$ are logically equivalent.

An *equational theory* is defined by a set E of equations. It induces a congruence on the term algebra, the least congruence $=_E$ on $\mathcal{T}$ that contains the term pairs $(\sigma s, \sigma t)$ for all $s = t \in E$ and all $\sigma \in \mathcal{SUB}$. An equational theory E together with the *standard equality axioms* – reflexivity, symmetry, transitivity, function and predicate replaceability for the symbols in Σ – is a definite clause theory, abbreviated by $E+$. A definite clause equality theory E also induces a least congruence $=_E$ on $\mathcal{T}$ corresponding to a model of the theory and the standard equality axioms (Jaffar et al. Maher 1984). We say that two terms s and t are *E-equal*, iff $s =_E t$. We extend E-equality to substitutions by $\sigma =_E \tau$, iff $\sigma x =_E \tau x$ for all variables $x \in \mathcal{V}$. We call two substitutions σ and τ E-equal on a set of variables $V \subseteq \mathcal{V}$, denoted $\sigma =_E \tau\, [V]$, when their restrictions to V are E-equal, i.e., when $\sigma v =_E \tau v$ for all $v \in V$. Notice, that an equational theory more exactly depends on two parameters, the signature Σ and the set of equations E, and that Σ must contain at least the function symbols occurring in E. Those function symbols of Σ that do not occur in E, are called *free* or *uninterpreted*. When $E = \emptyset$, we have the theory of syntactic equality (or *empty theory*). Notice, that different sets of equations may induce the same congruence relation on terms.

2.2 Unification

Given an equational or definite clause equality theory E and a system of equations Γ the solutions of the *E-unification problem* $\langle \Gamma \rangle_E$, also called the *E-unifiers of* Γ, are all substitutions σ with $DOM\sigma = Var(\Gamma)$, such that $\sigma s =_E \sigma t$ for all $s = t \in \Gamma$. The set of all E-unifiers of Γ is denoted by $U_E(\Gamma)$. For unification problems one is usually only interested in a subset of the set of all solutions that represents all E-unifiers. Representation is defined in terms of instantiation of substitutions on some set of variables W:

δ is an *E-instance on* W *of* σ ($\delta \geq_E \sigma\, [W]$) iff there is a λ with $\delta x =_E \lambda\sigma x$ for all $x \in W$.

If two substitutions are E-instances of each other on the same set of variables W, they are called *E-equivalent on* W.

The solution sets of unification problems are closed under E-instantiation on $Var(\Gamma)$ or any superset V of $Var(\Gamma)$: Every E-instance of an E-unifier is again an E-unifier. Hence we can define representative sets of E-unifiers or *complete sets of E-unifiers* $cU_E(\Gamma)$ by the property that the union of all E-instances of the elements of $cU_E(\Gamma)$ is exactly the set of all solutions $U_E(\Gamma)$: (let $V \supseteq Var(\Gamma)$)

(i) every $\delta \in cU_E(\Gamma)$ is an E-unifier (correctness)

(ii) for all $\delta \in U_E(\Gamma)$ exists $\sigma \in cU_E(\Gamma)$ such that $\delta \geq_E \sigma\, [V]$ (completeness)

If they exist, we are interested in *minimal* representative sets of E-unifiers (also called sets of *most general* E-unifiers), that are complete sets $\mu U_E(\Gamma)$ with the additional property:

(iii) for all $\sigma, \tau \in \mu U_E(\Gamma)$: $\sigma \geq_E \tau\, [W]$ implies $\sigma = \tau$ (minimality)

Notice that for equality theories, minimal sets of E-unifiers are unique up to E-equivalence on V (Fages & Huet 1986).

In order to keep proofs more readable we always require the following technical (separation) properties for complete sets of unifiers, whenever we deal with such complete solution sets. Every substitution σ of a complete set of unifiers has variable disjoint domain and codomain, i.e., $DOM\sigma \cap VCOD\sigma = \emptyset$. Every two elements σ, τ of a complete solution set have variable disjoint codomains ($VCOD\sigma \cap VCOD\tau = \emptyset$), and even more, when dealing with more than one complete set, we also require the elements of different complete sets to have pairwise variable disjoint codomains. These separation properties can always be obtained by renaming the codomains with fresh variables, respectively; each of these renamed substitutions is E-equivalent to the original substitution on V, and hence the sets of renamed substitutions are also complete sets of E-unifiers of the given problem (Bürckert et al. 1989).

Let us call a theory *E finitary*, if every E-unification problem has a finite, complete set of E-unifiers; we call it *unitary*, if for every E-unification problem there is a single E-unifier representing all solutions. Well known examples are the empty theory, which is unitary (Robinson 1967), and the theory of an associative and commutative function, which is finitary (Stickel 1975). Notice, that there also exist theories, where some systems of equations have only infinite complete sets of unifiers (e.g., the theory of an associative function, see Siekmann 1978) or even worse, for some systems no minimal sets of unifiers exist (e.g., a somewhat artificial theory in Fages & Huet 1986, or the theory of an associative and idempotent function, see Baader 1986 and Schmidt-Schauß

1986).

If we consider the definite clause theory $E+$ corresponding to an equational theory E, an E-unification problem is just a *query* to this logic program. The E-unifiers are the *correct answer substitutions* to this query (cf. Lloyd 1984). From the logical or model theoretical point of view a unification problem as well as a query to logic programs is corresponding to a formula, where all the variables are existentially quantified. The task to solve this problem is to prove this formula *constructively* that means to generate (all) witnesses for these existential variables that make the formula true (with respect to the theory).

3. Model Theory

In the following we use the common definitions of a structure or an algebra being a *model* for a formula or set of equations, respectively (cf. for example Shoenfield 1967 or Grätzer 1979). Notice, that an algebra is a model of a set of equations E iff the corresponding structure with equality is a model of the definite clause theory $E+$. Models whose domains are the set of ground terms, are also called *Herbrand models* (cf. for example Lloyd 1984 or Gallier 1986). We abbreviate as usual the fact that a structure $\mathcal{A}$ is a model of a formula F by $\mathcal{A} \models F$, and the fact that all models of a formula F are also models of another formula G by $F \models G$ – read G is a *consequence* of F or F *logically implies* G. Notice that a (first order, definite clause, equational) theory is usually defined as a set of formulae (clauses, equations) that are closed with respect to this consequence relation rather than as we did by a (not unique) set of axioms. However, the (unique) set of all consequences of these axioms is a theory in the common notion. For example the above E-equality congruence on the term algebra is the set of all consequences of the inducing set E of equations (this is essentially the completeness theorem for equational logic of Birkhoff 1935).

3.1 Solutions in a (Free) Algebra

In contrast to the logical task of solving equations and disequations with respect to all models of the given theory mentioned above, in mathematics a system of equations $s_i = t_i$ and disequations $p_j \neq q_j$ has to be solved in a single given algebra $\mathcal{A}$ (for instance, Diophantine equations have to be solved in the algebra of natural numbers or in the algebra of integers). We are searching for assignments of the variables X in the system with elements of $\mathcal{A}$ – or equivalently, for homomorphisms from the term algebra $\mathcal{T}(X)$ to the algebra $\mathcal{A}$ –, such that s_i and t_i are mapped to the same element, while p_j and q_j are mapped to different elements in $\mathcal{A}$. Model theoretically this is finding assignments of these variables that satisfy the corresponding existentially closed formula in the fixed model $\mathcal{A}$. Notice, that we can also replace $\mathcal{T}(X)$ by the term algebra $\mathcal{T}$ over infinitely many variables containing X.

Our definitions in section 2.2 correspond to this view, when we solve equations in the quotient $\mathcal{T}/{=_E}$ of the term algebra modulo a given equational theory. This algebra is (modulo isomorphism) the *E-free algebra* $\mathcal{F}_E$, that is the free algebra of the class of all models (the *variety*) of E (cf. Taylor 1979). It is well known that $\mathcal{F}_E$ is *generic* for E:

$$E \models \forall.s = t \qquad \text{iff} \quad \mathcal{F}_E \models \forall.s = t \qquad \text{(Tarski 1946).}$$

For definite clause equality theories genericity of $\mathcal{F}_E$ is a consequence of the fact that least Herbrand models are generic for definite clause theories (Lloyd 1984, Gallier 1986). For notational convenience, we shall represent these corresponding models identically. The distinctions between models and algebras are not significant in our treatment.

It might be convenient – although not really necessary – to clearly distinguish between the variables X occurring in the unification problem and the variables $\mathcal{V}$ used in the construction of the E-free algebra. This corresponds directly to our requirements for E-unifiers: For every assignment $\alpha\colon \mathcal{T}(X) \to \mathcal{F}_E$ solving a system of equations Γ with variables $X = Var(\Gamma)$ we can find a substitution σ with $DOM\sigma = X$, $VCOD\sigma \subseteq \mathcal{V}$, that is an E-unifier of the system, and corresponds to the assignment via $\alpha x = \kappa_E\sigma x = (\sigma x)/{=_E}$. Conversely, every E-unifier σ of the system corresponds an assignment $\alpha := \kappa_E\sigma$. Here κ_E is the *canonical* homomorphism of $\mathcal{T}$ onto $\mathcal{F}_E$.

3.1 Lemma: *a)* *A substitution σ solves an equation system Γ*
iff $\mathcal{F}_E \models \forall.\sigma\Gamma$
iff $\mathcal{F}_E \models \forall_X (\exists_{-X} [\sigma] \Rightarrow \Gamma)$.

b) *A set $cU_E(\Gamma)$ is a complete set of E-unifiers for a system Γ*
iff $\mathcal{F}_E \models \forall_X (\Gamma \Leftrightarrow \exists_{-X} [cU_E(\Gamma)]\,)$.

Proof: a) Let $\sigma s =_E \sigma t$ for all equations $s = t$ in Γ. Now let $\alpha\colon \mathcal{T}(W) \to \mathcal{F}_E$ be an assignment of $W = VCOD\sigma = Var(\sigma\Gamma)$. By the correspondence between such assignments and substitutions we have some δ with $\kappa_E\delta x = \alpha x$ for all $x \in VCOD\sigma$. Now $\delta\sigma s =_E \delta\sigma t$ for all $s = t$ in Γ, and by the universal mapping property $\kappa_E\delta r = \alpha r$ for all $r \in \mathcal{T}(W)$, and hence $\alpha\sigma s = \alpha\sigma t$ for all $s = t$ in Γ. The converse and the second equivalence are obvious.

b) "$\Rightarrow$" Because of part a) it is enough to prove that $\mathcal{F}_E \models \forall_X (\Gamma \Rightarrow \exists_{-X} [cU_E(\Gamma)]\,)$. Let $\alpha\colon \mathcal{T}(X) \to \mathcal{F}_E$ be any assignment with $\alpha s = \alpha t$ for each $s = t \in \Gamma$ in the E-free algebra. Then there is a substitution λ with $\alpha = \kappa_E\lambda$ and $\lambda s =_E \lambda t$ for each $s = t \in \Gamma$. By the completeness requirement there is some $\sigma \in cU_E(\Gamma)$ with $\lambda =_E \gamma\sigma\,[X]$ for some suitable substitution γ. Hence $\alpha = \kappa_E\gamma\sigma$ and we have constructed an assignment for $-X$ namely $\kappa_E\gamma$. This proves our claim.

"$\Leftarrow$" We show that $cU_E(\Gamma)$ is complete. Let λ be any E-unifier of Γ. Then $\kappa_E\lambda$ is an assignment satisfying $\kappa_E\lambda s = \kappa_E\lambda t$ for all $s = t \in \Gamma$ in the E-free algebra. Hence by assumption there is an assignment β of $-X$ and some $\sigma \in cU_E(\Gamma)$ with $\kappa_E\lambda = \beta\sigma$ and as again $\beta = \kappa_E\beta'$, we have $\lambda =_E \beta'\sigma\,[X]$. This proves completeness. ■

Notice, that by the genericity of the free algebra, we can replace $\mathcal{F}_E$ by E in part a) of the lemma. However, this is wrong for part b) as the following counterexample shows. Consider the unification problem $\langle f(x) = f(a)\,\rangle_\emptyset$ in the empty theory over the signature $\{f, a, b\}$ with a unary function symbol and two constants. Then obviously the substitution $\{x \leftarrow a\}$ is the only most general unifier. Now, take any algebra $\mathcal{A}$, where the two constants are interpreted different, $a^{\mathcal{A}} \neq b^{\mathcal{A}}$, but mapped to the same element by the operation $f^{\mathcal{A}}\colon f^{\mathcal{A}}(a^{\mathcal{A}}) = f^{\mathcal{A}}(b^{\mathcal{A}})$. This is obviously a model of the empty theory, but the equation has a solution $x = b^{\mathcal{A}}$ in this algebra that is not an "instance" of our unifier. This means that the completeness property is not "generic" for an equational theory, i.e., does not hold for all models of the theory.

In this paper we also solve disequations or systems of equations and disequations with respect to the E-free algebra. By the definition of the quotient this is equivalent to finding substitutions σ, such that the σs_i and σt_i are in the same equivalence class (i.e. $\sigma s_i =_E \sigma t_i$), while the σp_j and σq_j are in different equivalence classes (i.e. $\sigma p_j \neq_E \sigma q_j$). When solving equations every instance $\lambda\sigma$ of a solution σ is also a solution; in contrast this need not be the case for disequations. For instance, the substitution $\{x \leftarrow f(v)\}$ is a solution of the disequation $f(x) \neq f(f(a))$ under the empty theory since the terms $f(f(v))$ and $f(f(a))$ are syntactically different, its instance $\{x \leftarrow f(a)\}$, however, is of course not a solution. An immediate consequence is again in contrast to disequations: when an equation is solvable in $\mathcal{F}_E$, it is also solvable in every other model of E.

There are also applications (cf. Comon 1986, 1988, Lassez et al. 1987), where it is necessary to solve disequations with respect to a subalgebra of the E-free algebra, namely the *initial algebra* $\mathcal{I}_E$ *for E*, that is (modulo isomorphism) the E-quotient of the ground term algebra $\mathcal{T}_0$ (Goguen & Meseguer 1984). For equations this can equivalently be done by solving them over the E-free algebra (the solutions in the initial algebra are exactly the ground instances of the solutions in the E-free algebra), while for disequations this is again no longer the case. The following theorem shows that it is an equivalent task to solve an equation system with respect to all models of E or with respect to certain special models, the initial or the free algebra for E:

3.2 Theorem: *Let E be an equational or definite clause equality theory, and let Γ be a system of equations. Then the following are equivalent*

(i) $E \models \exists.\Gamma$
(ii) $\mathcal{I}_E \models \exists.\Gamma$
(iii) $\mathcal{F}_E \models \exists.\Gamma$

Proof: Since $\mathcal{I}_E$ is a model of E, we have that (i) implies (ii), and since $\mathcal{I}_E$ is a subalgebra (modulo isomorphism) of $\mathcal{F}_E$, we also have that (ii) implies (iii). By Lemma 3.1 we have that $\mathcal{F}_E \models \forall.\sigma\Gamma$. Since for every assignment α satisfying $\forall.\sigma\Gamma$ the composition $\alpha\sigma$ is an assignment satisfying $\exists.\Gamma$, the cycle is closed and all equivalences hold. ■

3.3 Corollary: *a)* $E \models \exists.\Gamma$ *iff there exist some* $\sigma \in \mathcal{SUB}(X, \mathcal{V})$ *with* $E \models \forall.\sigma\Gamma$
b) $\mathcal{F}_E \models \exists.\Gamma$ *iff there is a* $\sigma \in \mathcal{SUB}(X, \mathcal{V})$ *with* $\mathcal{F}_E \models \forall.\sigma\Gamma$
iff there is a $\Theta \subseteq \mathcal{SUB}(X, \mathcal{V})$ *with* $\mathcal{F}_E \models \forall_X (\exists_{-X} [\Theta] \Leftrightarrow \Gamma)$
c) $\mathcal{I}_E \models \exists.\Gamma$ *iff there exist some* $\sigma \in \mathcal{SUB}(X, \emptyset)$ *with* $\mathcal{I}_E \models \sigma\Gamma$
iff there exist some $\Theta \subseteq \mathcal{SUB}(X, \emptyset)$ *with* $\mathcal{I}_E \models [\Theta] \Leftrightarrow \Gamma$

Proof: a) + b) Follow immediately from Theorem 3.2 with Lemma 3.1.

c) Let $\mathcal{I}_E \models \exists.\Gamma$. Hence there is a ground substitution σ with $\mathcal{I}_E \models \sigma\Gamma$. Hence with Θ being the set of all ground solutions we have $\mathcal{I}_E \models [\Theta] \Leftrightarrow \Gamma$. The other direction is trivial. ■

Remarks: 1. A counterexample for disequations is the following: let $E := \{f(x) = c\}$ with the signature $\Sigma := \{c, f\}$, then $\mathcal{F}_E \models \exists.x \neq c$, but not $\mathcal{I}_E \models \exists.x \neq c$ and there is no substitution σ (over Σ!) with $\mathcal{F}_E \models \forall.\sigma x \neq \sigma c$.

Notice, that in this example $\mathcal{F}_E$ consists of the equivalence classes of the variables (they are singletons containing only the variable), and the equivalence class of the constant c containing all other terms, and hence $\mathcal{I}_E$ consists only of the equivalence class of c.

2. Theorem 3.2 says that for existentially closed equations both the initial and the free algebra are "generic", in contrast to universally closed equations, where only the free algebra is generic in general, but not the initial algebra. The theory of Remark 1 serves as a counterexample:

$$\mathcal{J}_E \models \forall x = c, \text{ but not } \mathcal{F}_E \models \forall x = c.$$

Thus for equations there is no difference between solving them in the special model $\mathcal{F}_E$, the E-free algebra, or solving them with respect to all models of E. For disequations, however, this is no longer the case. A reason for this is that an equational theory can only attempt to prove if two terms are equal. It has nothing to say about their inequality.

From the logical point of view equation solving is the task to prove constructively by giving witnesses for the variables that a system of equations Γ is a consequence of a certain equational theory E, more exactly the definite clause theory $E+$. To introduce inequality, we can restrict ourselves to fixed domains as above, or a weaker *closed world assumption* can be used, i.e., terms are not equal iff they cannot be proven equal in the theory. Below in section 3.3 we formalise this under the guise of the *equality completion* of a theory. This is an alternative – and as we will see later in some sense equivalent – approach to give semantics to the problem of solving disequations as the algebraic view pointed out above. Before, however, we must discuss, how the notion of solutions can be extended to more general theories.

3.2 Solutions in a (First Order) Theory

Let E be some set of formulae containing the standard equality axioms, that is, a first order theory with equality. Generalizing the view of the remark after Lemma 3.1, we can call a substitution σ an E-unifier of a system Γ of equations (an E-unification problem) iff $E \models \forall.\sigma\Gamma$. Notice, that for general theories such a solution need not exist. For example, let E be the theory $\{f(a) = c \vee f(b) = c\}$ and let $\Gamma := \{f(x) = c\}$. Then there is no single substitution for the variable x that solves this equation in that theory: Either $x = a$ or $x = b$ solves the equation, but we don't know, which one.

It can be seen directly that σ is an E-unifier of Γ, iff

$$E \models \forall([\sigma] \Rightarrow \Gamma)$$

or equivalently

$$E \models \forall_{Var(\Gamma)}(\exists_{-Var(\Gamma)}[\sigma] \Rightarrow \Gamma).$$

In the view of this we can reformulate instantiation by

$$\delta \text{ is an } \textit{E-instance of } \sigma \textit{ on } W \text{ iff } E \models \forall_W(\exists_{-W}[\delta] \Rightarrow \exists_{-W}[\sigma]),$$

and hence the definition of complete or minimal sets of E-unifiers in this more general framework is straight forward.

For equational or definite clause equality theories the two definitions of instantiation and hence of completeness or minimality are equivalent.

3.4 Lemma: *Given an equational or definite clause equality theory E and set of substitutions Θ, then*

$$\begin{array}{ll} & \delta \geq_E \theta \text{ for some } \theta \in \Theta\, [W] \\ \text{iff} & \mathcal{F}_E \models \forall_W (\exists_{-W} [\delta] \Rightarrow \exists_{-W} [\Theta]) \\ \text{iff} & E \models \forall_W (\exists_{-W} [\delta] \Rightarrow \exists_{-W} [\Theta]). \end{array}$$

Proof: Consider only the case for an equational theory, the proof is analogous with a definite clause equality theory. Consider, also, only the equivalence between the first and third condition, the equivalence of the second condition will fall out as a result. For some $\sigma \in \Theta$ and suitable λ we have

$$\delta x =_E \lambda \sigma x \text{ (for all } x \in W)$$

is equivalent to (note that $\delta x =_E \lambda x$ for all variables $x \in W\text{–}DOM\sigma$)

$$\mathcal{F}_E \models \bigvee_{\sigma \in \Theta} \forall . \exists_{-VCOD\delta - W} \bigwedge_{x \in W} \delta x = \delta \sigma x$$

is equivalent to

$$E \models \forall . \exists_{-VCOD\delta - W} \bigvee_{\sigma \in \Theta} \bigwedge_{x \in W} \delta x = \delta \sigma x.$$

(This last equivalence follows in the forward direction by genericity of the free algebra. The backwards direction follows via $\mathcal{F}_E \models \exists_{-VCOD\delta - W} \bigvee_{\sigma \in \Theta} (\bigwedge_{x \in W} \alpha \delta x = \alpha \delta \sigma x)$, where α is the assignment $\kappa_E\{x_i \leftarrow v_i\text{: } x_i \in Var(\delta W, W),\ v_i \text{ new distinct variables}\}$.)

This is equivalent to

$$E \models \forall . \bigvee_{\sigma \in \Theta} (\bigwedge_{x \in W} x = \delta x \Rightarrow \exists_{-W} \bigwedge_{x \in W} x = \sigma x),$$

in turn equivalent to

$$E \models \forall_W (\exists_{-W} [\delta] \Rightarrow \exists_{-W} [\Theta]),$$

we have finished the proof. ■

Surely the reader already recognized that there is no severe reason for restricting ourselves to queries of equations and to answers that are substitutions or equivalently the corresponding equations. We may allow more general formulae as queries and as answers. The first case happens, when we want to consider systems of equations *and* disequations, and we will see in the next sections that we also should loosen then our notion of answers, respectively.

3.3 Solutions under Equality Completion

As mentioned above we need a closed world assumption in order to handle negated equations, more exactly, the closed world assumption with respect to the equality predicate. A more general way than the algebraic view or "fixed algebra" semantics of section 3.1 is completion of the equality predicate (cf. Genesereth & Nilsson 1987). A succinct discussion for the motivation and theory behind this technique of equality completion – or unification complete theories – appears in (Jaffar & Lassez 1986).

Given a (first order) theory E, an *equality completion of E over Σ*, denoted E^C, is the universally quantified conjunction of the infinite set of axioms:

$$E^C := E + \cup \{\forall . (\Gamma \Rightarrow \exists_{-Var(\Gamma)} [\Theta_\Gamma])\text{: } \Gamma \text{ is an equation system over } \Sigma\}$$

where Θ_Γ is a complete set of E-unifiers of Γ. In the case where Θ_Γ is empty, the completion axiom $\forall . (\Gamma \Rightarrow \exists_{-Var(\Gamma)} [\Theta_\Gamma])$ corresponds to $\forall . \neg \Gamma$. When Θ_Γ contains the empty substitution ε the axiom can be ignored. Notice, that it is enough to range only over all "systems" Γ consisting of a single equation only. We sometimes may not explicitly mention the alphabet over which the completion is made. This must be done with care as the use of a different alphabet leads to a different theory. Both the E-free algebra $\mathcal{F}_E$ and the initial algebra $\mathcal{J}_E$ for E are models of the equality completion of an equational theory E (Theorem 3.2 and corollaries).

In the terminology of Jaffar et al. (1984) an equality completion of a definite clause pure equality theory E corresponds to the minimum set of axioms necessary for a theory to be a unification complete extension to the theory E. It can be seen that Clark's axioms (1978) for a unification complete counterpart of the syntactic equality theory are logically equivalent to the equality completion of this theory. His axioms are contained in the equality completion and they imply the equality completion (Jaffar & Stuckey 1986). Further examples of unification complete equality theories that correspond to equality completion with redundancy removed appear also in (Jaffar & Stuckey 1986).

A further advantage of this construction may lie in the following: when σ is an E-unifier of a system of equations, every "instance" of σ in a model $\mathcal{A}$ of E, i.e. the homomorhisms $\alpha\sigma$, where α is an assignment of the variables introduced by σ, is a solution of the system in this algebra $\mathcal{A}$. However, in general, not every solution in $\mathcal{A}$ can be factored by some E-unifier, for example there may be solutions in $\mathcal{A}$, but no E-unifiers (take $\mathcal{A}$ to be the trivial algebra containing only one element; see also the example after Lemma 3.1). For the models of an equality completion E^C, however, this always holds by definition.

3.5 Lemma: *Let $\mathcal{A} \models E^C$, and let Γ be a set of equations. Then an assignment α of the variables of Γ with elements of $\mathcal{A}$ solves the equations iff there is an E-unifier σ of Γ and an assignment β of the variables introduced by σ, such that $\alpha = \beta\sigma$ on the variables of Γ.*

Proof: One direction is obvious. For the other direction by the definition of E^C there is an E-unifier σ in Θ_Γ and an assignment β such that $\alpha\Gamma \Rightarrow \alpha x_1 = \beta\sigma x_1 \wedge \ldots \wedge \alpha x_n = \beta\sigma x_n$ is true in $\mathcal{A}$, where $x_1,\ldots, x_n$ are the variables of Γ. Since α solves Γ in $\mathcal{A}$, this completes the proof. ■

As the definition of an equality completion depends on the chosen complete sets of E-unifiers, we need to prove that different equality completions are logically equivalent.

3.6 Theorem: *For an equational or definite clause theory E, any equality completion of E over some signature is satisfiable and unique (in the sense that it is logically equivalent to every other completion over the same signature).*

Proof: 1. Let $\mathcal{M}$ be a Herbrand model of E. We must prove that $\mathcal{M} \models \forall.\Gamma \Rightarrow [\Theta_\Gamma]$. Since $\mathcal{M}$ is a Herbrand model, it is enough to show that, for every ground substitution γ of the variables in Γ, if $\gamma\Gamma$ is true in $\mathcal{M}$, then also $\gamma[\Theta_\Gamma]$ is true in $\mathcal{M}$. But this is obviously the case, since, when γ is an E-unifier of Γ, it must be an E-instance of some $\theta \in \Theta_\Gamma$.

2. We show that any equality completion of E has the formula $\forall_V (\exists_{-V}[\Theta_\Gamma] \Rightarrow \exists_{-V}[\Psi])$ with $V = Var(\Gamma)$ as logical consequence, if Θ_Γ is the complete set of E-unifiers for Γ chosen in the given equality completion and Ψ is any complete set of E-unifiers for Γ. As

Ψ is complete, there exists a $\psi \in \Psi$ for each $\theta \in \Theta_\Gamma$ with $E \models \forall_V (\exists_{-V} \theta \Rightarrow \exists_{-V} \psi)$ and hence also $E^C \models \forall_V (\exists_{-V} \theta \Rightarrow \exists_{-V} \psi)$. Thus we have $E^C \models \forall_V (\exists_{-V} [\Theta_\Gamma] \Rightarrow \exists_{-V} [\Psi])$, and by the definition of the equality completion $E^C \models \forall_V (\Gamma \Rightarrow \exists_{-V} [\Psi])$. Therefore any two equality completions must be equivalent. ■

Together with Lemma 3.1 and Lemma 3.4 we have as an immediate corollary that for an equational or definite clause equality theory E the E-free algebra and the equality completion approach are equivalent in the following sense:

3.7 Corollary: *A set $cU_E(\Gamma)$ is a complete set of E-unifiers of Γ*
iff $\mathcal{F}_E \models \forall.(\Gamma \Leftrightarrow \exists_{-Var(\Gamma)} [cU_E(\Gamma)])$
iff $E^C \models \forall.(\Gamma \Leftrightarrow \exists_{-Var(\Gamma)} [cU_E(\Gamma)])$.

As we will see in the next section both solving *disequations* in E-free algebras or solving under equality completion are also equivalent with respect to complete sets of solutions in this sense, provided we generalize our notions of solutions. While the E-free algebra approach fits into the mathematical view of solving equations or disequations in a certain algebra, the equality completion approach shows the logical aspect of this task by a more general closed world assumption, moreover it supports generalization to arbitrary first order theories.

4. Disunification Problems

Let us denote the problem to resolve a system $\Gamma \cup \Delta$ of equations and disequations over $\mathcal{F}_E$ as an *E-disunification problem*, written

$$\langle \Gamma, \Delta \rangle_E := \langle s_i = t_i \colon 1 \le i \le n, p_j \neq q_j \colon 1 \le j \le m \rangle_E.$$

An *E-solution* of $\langle \Gamma, \Delta \rangle_E$ is any substitution $\sigma \in \mathcal{SUB}(V, \mathcal{V})$, such that $\sigma s_i =_E \sigma t_i$ for $1 \le i \le n$ and $\sigma p_j \neq_E \sigma q_j$ for $1 \le j \le m$, where $V = Var(\Gamma, \Delta)$. The set of all these solutions is denoted $S_E(\Gamma, \Delta)$.

As already discussed, the solutions to such problems unfortunately cannot be represented by the same instantiation method used for pure unification problems. The reason is that solution sets are no longer closed under instantiation. For example, the disunification problem $\langle x = y, x \neq a \rangle_\emptyset$ under the syntactic theory has a solution $\{x \leftarrow v, y \leftarrow v\}$, but the instance $\{x \leftarrow a, y \leftarrow a\}$ is not a solution.

4.1 Generalized Solutions

A way out of this trouble is to define more general notions of solutions (cf. Comon 1986, Lassez et al. 1987, Lescanne & Kirchner 1987, Smolka et al. 1987). As we have seen, a substitution represents the set of all its instances, hence in order to describe all instances *except* certain ones we propose *differences* of substitutions mirroring the difference of the corresponding instance sets. For example all solutions of the disunification problem $\langle f(x, g(u)) = f(y, y), x \neq g(a) \rangle_\emptyset$ for the variables in $V = \{x,y,u\}$ can be represented by the difference of the substitutions $\{x \leftarrow g(u), y \leftarrow g(u)\}$ and $\{x \leftarrow g(a)\}$. Thus we define a *substitution with exceptions* on a domain V as a pair $\sigma - \Psi$ of a substitution $\sigma \in \mathcal{SUB}(V, \mathcal{V})$ and a family of substitutions $\Psi = \{\psi_\iota \in \mathcal{SUB}(V, \mathcal{V}) \colon \iota \in I\}$, the *exceptions.* Another way to represent the set of all instances of a substitution σ except certain ones is to classify the exceptions by constraints on the allowed instantiations of the free variables present after the substitution has been applied (that is, those in $Var(\sigma V)$). A solution to the above roblem then is $\{x \leftarrow g(u), y \leftarrow g(u)\}$ except when $\{u \leftarrow a\}$. That is we consider a *constrained substitution* on a domain V as a pair $\sigma /\!/ \Theta$ of a substitution $\sigma \in \mathcal{SUB}(V, \mathcal{V})$ and a family of substitutions $\Theta = \{\theta_\iota \in \mathcal{SUB}(Var(\sigma V), \mathcal{V}) : \iota \in I\}$, the *constraints*. In the following we only will consider substitutions with exceptions (constrained substitutions) on V where the codomains of all of the occurring substitutions are pairwise variable disjoint except for variables in V ($Var(\sigma V)$).

Whereas we can treat a substitution σ as a conjunction of equations, $[\sigma]$, we can treat a substitution with exceptions $\sigma - \Psi$ on V (or a constrained substitution $\sigma /\!/ \Theta$) as a conjunction whose parts are equations that correspond to σ and disjunctions of disequations that each correspond to one of the negated substitution from Ψ (or Θ), more exactly we have the corresponding formula $[\sigma] \wedge \neg\exists_{-V}[\Psi]$ ($[\sigma] \wedge \neg\exists_{-Var(\sigma V)}[\Theta]$). Care must be taken with the variable quantification because the codomain variables of the exceptions and the constraints are universally quantified. We cover this in more detail in section 4.2.

Unfortunately, not every solution of a disunification problem will have finitely many exceptions. Let $\langle x = y, a.x \neq x.a \rangle_A$ be a disunification problem with respect to an associative function ".". Solutions to this problem are all substitutions of x and y with the same arbitrary string except the infinitely many strings $aa...a$ of length n (for short a^n), for all $n \geq 1$. They can be represented by the following substitution with exceptions or constrained substitution:

$$\{x \leftarrow v, y \leftarrow v\} - \{\{x \leftarrow a^n\} : n \geq 1\} \text{ or } \{x \leftarrow v, y \leftarrow v\} /\!/ \{\{v \leftarrow a^n\} : n \geq 1\}.$$

It is useful to consider both ways of representing instances. While the first form is more suitable for testing whether or not a given substitution is an instance, the second form better supports generation of instances (of course provided there are only finitely many exceptions or constraints, cf. definition below). Beyond these two forms, a mixture of them or even a weaker concept of representation may be useful. One could just solve the equations and keep the disequations themselves as constraints without solving:

A pair$(\sigma, \sigma\Delta)$ "solves" $\langle \Gamma, \Delta \rangle_E$, when σ is an E-unifier of Γ.

In order to extend the notions of instances of substitutions to our more general solved forms, we have in mind a picture of a substitutions as a representative set of all its instances. A substitution with exceptions then represents the instances of its substitution part minus those of its exceptions (and similarily for constrained substitutions):

$$Instances(\sigma - \Psi) = Instances(\sigma) \setminus \bigcup_{\psi \in \Psi} Instances(\psi).$$

4.1 Definition: Let W be any set of variables, and let us say a substitution λ is an *E-instance on W* of a set of substitutions Ψ (abbreviated $\lambda \geq_E \Psi\,[W]$) iff any instance of λ is an instance of some $\psi \in \Psi$. Then:

1. A substitution λ is an *E-instance on W* of a substitution with exceptions $\sigma - \Psi$ (abbreviated by $\lambda \geq_E \sigma - \Psi\,[W]$), iff λ is an E-instance on W of σ, but not of Ψ.

2. A substitution λ is an *E-instance on W* of a constrained substitution $\sigma /\!/ \Theta$ (abbreviated by $\lambda \geq_E \sigma /\!/ \Theta\,[W]$), iff there is some γ with $\lambda =_E \gamma\sigma\,[W]$, which is not an E-instance on $Var(\sigma W)$ of Θ.

We call a substitution with exceptions or a constrained substitution *E-consistent on W*, iff it has at least one E-instance on W, otherwise we call it *E-inconsistent on W*. ∎

Obviously the two instance notions are consistent with the common one for substitutions as defined above, that is, $\lambda \geq_E \sigma\, [W]$ iff λ is an instance of σ, where σ is considered as a substitution with (empty) exceptions or constraints or as a singleton set of substitutions.

Notice that we could have just as well defined $\lambda \geq_E \Psi\, [W]$ by "λ is an instance of some substitution $\psi \in \Psi$". This definition, however, causes problems if we wish to work in other algebras or models. For example, in the initial algebra semantics where the signature contains only the constant a and the unary function symbol f, the set of solutions for x represented by the substitution $\{x \leftarrow f(y)\}$ is exactly identical to the set of solutions represented by the set $\{\{x \leftarrow f(a)\}, \{x \leftarrow f(f(y))\}\}$, but the former substitution is not an instance of any substitution in the latter set.

These instance notions can of course be interpreted logically in terms of the free algebra. We outline these simple results because they form the basis of much of our subsequent development. Notice that instantiation and solutions relative to a particular model or algebra could have been defined in these terms.

4.2 Lemma: *Let λ be a substitution.*

a) λ is an E-instance on W of a set of substitutions Ψ
iff $\kappa_E\lambda$ satisfies $\exists_{-V}\, [\Psi]$ in $\mathcal{F}_E$.

b) λ is an E-instance on W of a substitution with exceptions $\sigma - \Psi$
iff $\kappa_E\lambda$ satisfies $\exists_{-V}[\sigma] \wedge \neg\exists_{-V}[\Psi]$ in $\mathcal{F}_E$.

c) λ is an E-instance on W of constrained substitution $\sigma \| \Theta$
iff $\kappa_E\lambda$ satisfies $\exists_{-V}([\sigma] \wedge \neg\exists_{-Var(\sigma V)}[\Theta])$ in $\mathcal{F}_E$.

d) λ solves a disunification problem $\langle \Gamma, \Delta \rangle_E$
iff $\kappa_E\lambda$ satisfies $\Gamma \wedge \Delta$ in $\mathcal{F}_E$.

Proof: a) This follows similarly to Lemma 3.4.

b–d) These follow directly from the definition and part a). ∎

Inconsistency

While a substitution always has instances, this may no longer be the case for substitutions with exceptions and constrained substitutions: $\{x \leftarrow f(v)\} - \{\{x \leftarrow f(w)\}\}$ or equivalently $\{x \leftarrow f(v)\} \| \{\{v \leftarrow w\}\}$ have no instances because in the first form the substitution can be an instance of one of its exceptions and in the second form the constraints span the whole set of substitutions. The lemma below shows how to detect inconsistency, and notice that it applies in any model/algebra of E^C when using instantiation within the model/algebra.

4.3 Inconsistency Lemma: *Let W a set of variables.*

1. A substitution with exceptions $\sigma - \Psi$ has no E-instances on W
iff σ is an E-instance on W of Ψ.

2. A constrained substitution $\sigma \| \Theta$ has no E-instances on W
iff ε (the identity) is an E-instance on $Var(\sigma W)$ of Θ.

Proof: 1. If σ is an instance of Ψ then all instances of σ are instances of Ψ. Hence there is no instance of $\sigma-\Psi$. Conversely, assume λ is an instance of $\sigma-\Psi$ and σ is an instance of Ψ then λ is an instance of Ψ, a contradiction.

2. If ε is an instance of Θ then all substitutions are instances of Θ. Hence $\lambda =_E \gamma\sigma\,[W]$, where γ is not an instance of Θ is impossible, that is, there exists no instance of $\sigma /\!/ \Theta$. Conversely, assume there is an instance $\lambda =_E \gamma\sigma\,[W]$ of $\sigma /\!/ \Theta$ and ε is an instance of Θ then γ is also an instance of Θ, again a contradiction. ■

4.4 Corollary: *A substitution with exceptions or constrained substitution has no E-instances on W in any model/algebra of E^C if it has no instances in the E-free algebra.*

Proof: This follows directly from Lemma 3.4. ■

From the computational point of view, our solved forms will only make sense if we can decide whether or not they are consistent. In the free algebra, sufficient conditions are that there are at most finitely many exceptions or constraints and that we can decide the problem of whether a substitution σ is an E-instance on W of another substitution φ. The latter decision can be done by solving the E-unification problem $\langle\, \sigma x = \varphi x\!: x \in W\, \rangle_E$, where the variables of φx are treated as constants, provided we have a suitable E-unification algorithm. Unification problems, where one side is (considered) ground, are known as *E-matching* problems (cf. Bürckert 1989 for a discussion of the relationships between E-matching and E-unification). In this situation we also can decide redundancy of exceptions or constraints.

If we want to deal with the initial algebra semantics, we have to test whether or not a substitution with exceptions or a constrained substitution has ground instances ("ground consistency"). Hence by the Inconsistency Lemma, we need a decision procedure for testing if all ground instances of a substitution are E-instances of a (finite) set of substitutions. An example will demonstrate what may happen: The two substitutions $\{v \leftarrow a\}$ and $\{v \leftarrow f(w)\}$ span *all* ground substitutions (with respect to the variable set $\{v\}$), if we suppose that there are only the constant a and the unary function symbol f in the signature That is, the constrained substitution $\{x \leftarrow v\} /\!/ \{\{v \leftarrow a\}, \{v \leftarrow f(w)\}\}$ has no ground instances because (using the Inconsistency Lemma) ε is an instance of $\{\{v \leftarrow a\}, \{v \leftarrow f(w)\}\}$ in the initial algebra – in contrast with the free algebra.

Translating between Representations

How do the two representations, exceptions or constraints, compare? One way of answering this question is to show how translations can be made between them. The notions of instantiation allow such translations. Notice that while the lemma below could have been developed in terms of the free algebra $\mathcal{F}_E$, we have instead worked in the logical context via Lemma 4.2. A corresponding Translation Lemma, then, applies in *any* model or algebra of E^C. For example, the lemma holds for E-instances in the initial algebra as well.

4.5 Translation Lemma: *The substitution E-instances on W of the substitution with exceptions $\sigma-\Psi$ are equivalent to those of the constrained substitution $\sigma \| \bigcup_{\psi\in\Psi} cU_E(\sigma w = \psi w: w\in W)$.*
2. Let $(\Theta\sigma)/_W$ represent $\bigcup_{\theta\in\Theta}\{(\theta\sigma)/_W\}$. If E is an equational or definite clause equality theory such that for any term t and substitution τ, $cU_E(t = \tau t) = \{\tau\}$, then the substitution E-instances on W of the constrained substitution $\sigma\|\Theta$ are equivalent to those of the substitution with exceptions $\sigma-(\Theta\sigma)/_W$.

Proof: Notice that $\exists_{-W}[\sigma] \wedge \neg\exists_{-W}[\Psi] \Leftrightarrow \exists_{-W}([\sigma] \wedge \neg\exists_{-Var(\sigma W)} \bigvee_{\psi\in\Psi}\bigwedge_{w\in W} \sigma w=\psi w)$, since $W \supseteq Var(\sigma)\cap Var(\Psi)$. And $\bigvee_{\psi\in\Psi}\bigwedge_{w\in W} \sigma w=\psi w \Leftrightarrow [\bigcup_{\psi\in\Psi} cU_E(\sigma w=\psi w: w\in W)]$, since we are in a model of E^C. In the free algebra, the result follows from Lemma 4.2. ■

Let us call a theory E Ω-free (cf. Szabo 1982, Bürckert et al. 1989), if for all function symbols f of the signature

$$f(s_1,\ldots,s_n) =_E f(t_1,\ldots,t_n) \text{ implies } s_i =_E t_i \ (1 \le i \le n).$$

Then the following corollary to the Translation Theorem holds.

4.6 Corollary: *Let $(\Theta\sigma)/_W$ represent $\bigcup_{\theta\in\Theta}\{(\theta\sigma)/_W\}$. If E is an Ω-free equational or definite clause equality theory, then the substitution E-instances on W of the constrained substitution $\sigma\|\Theta$ are equivalent to those of the substitution with exceptions $\sigma-(\Theta\sigma)/_W$.*

Proof: From the Translation Theorem we get that the substitution E-instances of $\sigma-(\Theta\sigma)/_W$ are equivalent to those of $\sigma\|\bigcup_{\theta\in\Theta} cU_E(\sigma w=\theta\sigma w: w\in W)$. It remains to be shown that for every $\theta\in\Theta$, θ is a most general unifier for $\langle \sigma w=\theta\sigma w: w\in W \rangle_E$. Obviously θ is an E-unifier (idempotency of θ). By the Ω-freeness property we have that every solution of the problem is an instance of θ: $\lambda\sigma w =_E \lambda\theta\sigma w$ for all $w\in W$) implies $\lambda x =_E \lambda\theta x$ for all $x \in Var(\sigma W)$, and the idempotency of θ implies $\lambda x =_E \lambda\theta x$ for all variables $x \in Var(\theta\sigma W)$. Hence λ is an E-instance of θ. ■

Consider the equational theory E given by $\{f(a, x) = f(a, y)\}$ over the signature containing only the constant symbol a and the binary function symbol f. First, if we ignore the warning concerning equality theories given in part 2 of the Translation Lemma, we would translate the constrained substitution $\{x \leftarrow f(a, u)\} \| \{u \leftarrow a\}$ into the substitution with exceptions $\{x \leftarrow f(a, u)\}-\{x \leftarrow f(a, a)\}$ which due to E is inconsistent. But notice the constrained substitution also simplifies to $\{x \leftarrow f(a, u)\}$ (u can take on any value), so the lemma is clearly not applicable for this equality theory.

Redundancy

The form of substitutions with exceptions or constrained substitutions given can sometimes be further simplified. There may be *redundant* exceptions or constraints, that is, if we remove them, we will have exactly the same instances as before. For example, in the context of the empty equational theory, consider the following substitution with exceptions on $\{x, y\}$:

$$\{x \leftarrow f(u, v), y \leftarrow a\} - \{\{x \leftarrow f(g(z), b)\}, \{x \leftarrow f(g(c), b)\}, \{x \leftarrow f(w, b), y \leftarrow d\}\}.$$

This can be simplified to

$$\{x \leftarrow f(u,v), y \leftarrow a\} - \{\{x \leftarrow f(g(z), b)\}\}.$$

Even worse, consider the following two substitution with exceptions on $\{x\}$:

$$\{x \leftarrow f(u)\} - \{\{x \leftarrow f(f(v))\}\} \quad \text{and} \quad \{x \leftarrow f(f(v))\}.$$

In this case the effect of the exception is negated by the second substitution, and together they simplify to

$$\{x \leftarrow f(u)\}.$$

In our experience with an experimental interpreter for logic programs using syntactic equality and constrained substitutions, these forms of redundancy and others besides were common. The identification of redundant constraints, exceptions or substitutions is a key problem as it can significantly simplify a solution's representation, as well as subsequent computation making use of the solution.

The following six cases of redundant exceptions may arise:

Case I: An exception ψ_0 is an instance of the other exceptions, that is, ψ_0 is an "E-merge" of a subset of $\Psi \setminus \{\psi_0\}$ (E-merges can be computed by E-unification: an E-merge of a set of substitutions is any E-unifier of the conjunction of the equational representations of those substitutions, cf. Herold 1987 for more details and some results on E-merging). This is the case for the second exception in the first example above.

Case II: An exception ψ_0 and the substitution σ have no common E-instances, that is, they have no E-merge. This is the case for the third exception in the first above.

Case III: As a combination of the above two cases, some instances of ψ_0 are instances of $\Psi \setminus \{\psi_0\}$ and the others are not instances of σ.

Case IV: Another substitution with exceptions in the set covers those instances that would be excluded by an exception ψ_0. This is the case in the second example above.

In addition, the following two cases of redundant substitutions with exceptions may arise:

Case V: A substitution with exceptions is a special case of one another in the set.

Case VI: A substitution with exceptions is a special case of several others in the set.

To make these forms more explicit we need to define, first, what it means for substitutions with exceptions and constrained substitutions to be free of redundancy, and second, how relative instantiation of them can be compared. Instantiation is used as a tool to detect redundancy. The notion of "redundancy-free" is captured by the definition of reduced below; this is analogous to the definition for reduced definite clauses used in Generalised Subsumption (Buntine 1988).

4.7 Definition: A set of substitutions with exceptions $\{\sigma_\iota - \Psi_\iota : \iota \in I\}$ on V is *reduced* (or *minimal*) if, for any $\{\sigma_\iota - \Psi'_\iota : \iota \in K\}$ produced by removing at least some exceptions or at least some substitutions with exceptions from $\{\sigma_\iota - \Psi_\iota : \iota \in I\}$ and $I \supseteq K$, there exists a substitution λ and an $\iota \in I$ such that $\lambda \geq_E \sigma_\iota - \Psi_\iota\ [V]$ but no $\kappa \in K$ such that $\lambda \geq_E \sigma_\kappa - \Psi'_\kappa\ [V]$. ■

We call the process of removing an exception or a substitution with exceptions from a set of substitutions with exceptions *reduction* just when the substitution E-instances of the set remain unchanged by the removal. So if we cannot *reduce* a set of substitutions with exceptions any further, they must be *reduced.*

The definition of reduced for constrained substitutions is equivalent. Notice that a single substitution with exceptions is reduced if and only if no exception can be removed from it. The following definitions of instantiation are a natural extension of the previous.

4.8 Definition: Let W be any set of variables. We call a substitution with exceptions $\delta\text{–}\Lambda$ an *E-instance on W* of a substitution with exceptions $\sigma\text{–}\Psi$ iff $\lambda \geq_E \delta\text{–}\Lambda\ [W]$ implies $\lambda \geq_E \sigma\text{–}\Psi\ [W]$. We abbreviate this by $\delta\text{–}\Lambda \geq_E \sigma\text{–}\Psi\ [W]$.

We call a substitution with exceptions $\delta\text{–}\Lambda$ an *E-instance on W* of a set of substitutions with exceptions $\{\sigma_\iota\text{–}\Psi_\iota\text{: } \iota \in I\}$ iff $\lambda \geq_E \delta\text{–}\Lambda\ [W]$ implies $\lambda \geq_E \sigma_\iota\text{–}\Psi_\iota\ [W]$ for some index ι. We abbreviate this by $\delta\text{–}\Lambda \geq_E \{\sigma_\iota\text{–}\Psi_\iota\text{: } \iota \in I\}\ [W]$. ∎

Again, the definitions of instantiation for constrained substitutions are equivalent. So a substitution with exceptions can be reduced from a set of substitutions with exceptions if it is an E-instance of the remaining substitutions with exceptions in the set. This notion of instantiation has a slightly different flavour to that found for common substitutions. For example, we showed above that $\{x \leftarrow f(u)\}$ is an E-instance on $\{x\}$ of

$$\{x \leftarrow f(u)\} - \{\{x \leftarrow f(f(v))\}\} \quad \text{and} \quad \{x \leftarrow f(f(v))\},$$

but notice that it is *not* an E-instance of either of these substitutions taken individually.

The following lemma shows how such E-instances can be detected recursively. This allows redundancy cases V and VI to be detected.

4.9 Instantiation Lemma: *1. An E-consistent (on W) substitution with exceptions $\delta\text{–}\Lambda$ is an E-instance on W of a set of substitutions with exceptions $\{\sigma_\iota\text{–}\Psi_\iota\text{: } \iota \in I\}$ (where $1 \in I$) iff*

$$\delta \geq_E \sigma_1\ [W] \;\; or \;\; \delta\text{–}(\Lambda \cup \{\sigma_1\}) \geq_E \{\ \sigma_\iota\text{–}\Psi_\iota\text{: } \iota \in I\text{–}\{1\}\ \}\ [W],$$

and for some $\psi \in \Psi_1$ and each $\rho \in cU_E(\delta w = \psi w\text{: } w \in W)$

$$\rho\delta \geq_E \Lambda\ [W] \;\; or \;\; \rho\delta\text{–}\Lambda \geq_E \{\ \sigma_\iota\text{–}\Psi_\iota\text{: } \iota \in I\text{–}\{1\}\ \}\ [W].$$

2. Let E satisfy the conditions in part 2 of the Translation Lemma. An E-consistent (on W) constrained substitution $\delta \,\|\, \Lambda$ is an E-instance on W of a set of constrained substitutions $\{\sigma_\iota \,\|\, \Theta_\iota\text{: } \iota \in I\}$ (where $1 \in I$) iff

$$\delta \geq_E \sigma_1\ [W] \;\; or \;\; \delta \,\|\, (\Lambda \cup cU_E(\delta w = \sigma_1 w\text{: } w \in W)) \geq_E \{\sigma_\iota \,\|\, \Theta_\iota\text{: } \iota \in I\text{–}\{1\}\}\ [W],$$

and for some $\theta \in \Theta_1$ and every $\rho \in cU_E(\delta w = \theta\sigma_1 w\text{: } w \in W)$

$$\rho\delta \geq_E \Lambda\delta\ [W] \;\; or \;\; \rho\delta \,\|\, \bigcup_{\lambda \in \Lambda} cU_E(\rho\delta w = \lambda\delta w\text{: } w \in W\ ) \geq_E \{\sigma_\iota \,\|\, \Theta_\iota\text{: } \iota \in I\text{–}\{1\}\}[W].$$

Proof: 1. The left-hand side by definition is equivalent to: $\lambda \geq_E \delta\ [W]$ and not $\lambda \geq_E \Lambda\ [W]$ implies there exists $\iota \in I$ such that $\lambda \geq_E \sigma_\iota\ [W]$ and not $\lambda \geq_E \Psi_\iota\ [W]$ iff $\lambda \geq_E \delta\ [W]$ and not $\lambda \geq_E \Lambda\ [W]$ and not $\lambda \geq_E \sigma_1 [W]$ *or* $\lambda \geq_E \delta\ [W]$ and not $\lambda \geq_E \Lambda\ [W]$ and $\lambda \geq_E \Psi_1 [W]$ implies there exists an $\iota \in I\text{–}\{1\}$ such that $\lambda \geq_E \sigma_\iota\ [W]$ and not $\lambda \geq_E \Psi_\iota\ [W]$. Note that as in the proof of the Translation Lemma part 1, $\lambda \geq_E \delta\ [W]$ and $\lambda \geq_E \Psi_1 [W]$ iff $\lambda \geq_E \rho\delta\ [W]$ for some $\rho \in cU_E(\delta w = \psi w\text{: } w \in W)$ for some $\psi \in \Psi_1$, and the result follows again by definition and the Inconsistency Lemma.

2. This follows from part 1 and the Translation Lemma. ∎

The Instantiation Lemma does have some simpler special cases. When comparing a single substitution against another, we get:

4.10 Corollary: *1. A substitution with exceptions $\delta-\Lambda$ is an E-instance on W of a substitution with exceptions $\sigma-\Psi$ iff $\delta \geq_E \sigma$ [W] and there is some $\psi \in \Psi$, such that for every $\rho \in cU(\delta w = \psi w\colon w \in W)$, $\rho\delta \geq_E \Lambda$ [W].*
2. Let E satisfy the conditions of part 2 of the Translation Lemma. A constrained substitution $\delta \| \Lambda$ is an E-instance on W of a constrained substitution $\sigma \| \Theta$ iff $\delta \geq_E \sigma$ [W] and there is a $\theta \in \Theta$, such that for each $\rho \in cU_E(\delta w = \theta\sigma w\colon w \in W)$, $\rho\delta \geq_E \Lambda\delta$ [W].

The Instantiation Lemma also yields results for detecting redundant exceptions or constraints, and so allows redundancy cases I-IV to be detected.

4.11 Corollary: *1. The exception $\psi_0 \in \Psi_1$ can be reduced from Ψ_1 in the set of substitutions with exceptions $\{\sigma_\iota - \Psi_\iota\colon \iota \in I\}$ on W (where $1 \in I$) iff for each $\rho \in cU_E(\sigma_1 w = \psi_0 w\colon w \in W)$*

$$\rho\sigma_1 \geq_E \Psi_1 \setminus \{\psi_0\} \; [W]$$

or

$$\rho\sigma_1 - \Psi_1 \setminus \{\psi_0\} \geq_E \{\, \sigma_\iota - \Psi_\iota \colon \iota \in I - \{1\} \,\} \; [W].$$

2. Let E satisfy the conditions in part 2 of the Translation Lemma. The constraint $\theta_0 \in \Theta_1$ can be reduced from Θ_1 in the set of constrained substitutions $\{\sigma_\iota \| \Theta_\iota\colon \iota \in I\}$ on W (where $1 \in I$) iff

$$\theta_0\sigma_1 \geq_E \Theta_1\sigma_1 \setminus \{\theta_0\sigma_1\} \; [W]$$

or

$$\theta_0\sigma_1 \| \bigcup_{\theta \in \Theta_1 \setminus \{\theta_0\}} cU_E(\theta_0\sigma_1 w = \theta\sigma_1 w\colon w \in W) \geq_E \{\sigma_\iota \| \Theta_\iota\colon \iota \in I - \{1\}\} \; [W].$$

As an example of Corollary 4.11 Part 2, let $x,u,v \in \mathcal{V}$ and let E be given by the equation $\{f(x, a) = f(y, a)\}$, and consider the single constrained substitution $\sigma \| \Theta$ on $W = \{x\}$ given by $\{x \leftarrow f(u,v)\} \| \{\, \{u \leftarrow b, v \leftarrow a\}, \{u \leftarrow c, v \leftarrow a\} \,\}$. Suppose we wish to apply the corollary with θ_0 given by $\{u \leftarrow b, v \leftarrow a\}$. The lemma says to reduce θ_0 iff $\theta_0\sigma \geq_E \Theta\sigma \setminus \{\theta_0\sigma\}$ [W]. Clearly this holds true. Notice that if we apply the Translation Lemma it can be readily seen that θ_0 should reduce. Notice also, that a candidate simplification of Corollary 4.11, reduce θ_0 iff $\theta_0 \geq_E \Theta \setminus \{\theta_0\}$ [Var(σW)] does *not* correctly predict the reduction.

Complete Representations

As with substitutions we are interested in representative sets of substitutions with exceptions or constrained substitutions. Therefore we define a set of substitutions with exceptions S to be a complete representation for the solutions of a disunification problem $\langle \Gamma, \Delta \rangle_E$, iff the instances of the elements of S are exactly the solutions of the disunification problem (again, where $V = Var(\Gamma, \Delta)$):

(i) $\lambda \geq_E \sigma - \Psi$ [V] for some $\sigma - \Psi \in S$ implies λ solves $\langle \Gamma, \Delta \rangle_E$ (correctness)

(ii) λ solves $\langle \Gamma, \Delta \rangle_E$ implies $\lambda \geq_E \sigma - \Psi$ [V] for some $\sigma - \Psi \in S$ (completeness)

Usually such a representative set should not contain inconsistent elements, we require:

(iii) $\sigma - \Psi$ is E-consistent on V for all $\sigma - \Psi \in S$ (consistency)

A final property, possibly pertaining to a representative set is minimality or reduction; this was described for substitutions with exceptions in the previous section. Representative sets of constrained substitutions are defined analogously. Obviously these definitions are again consistent with the corresponding definitions for common substitutions; correctness, completeness and minimality for a set of substitutions are equivalent to correctness, completeness and minimality for the set of substitutions considered as set of substitutions with (empty) exceptions or constraints.

Again we call a theory *finitary* (w.r.t disunification), if every disunification problem has a finite complete set of substitutions with exceptions or constrained substitutions representing all its solutions; and we call it *unitary* (w.r.t. disunification), if the solutions of every disunification problem can be represented by a single substitution with exceptions or constrained substitution.

The logical interpretation of correctness and completeness is as follows.

4.12 Corollary to Lemma 4.2: *Let* $\langle \Gamma, \Delta \rangle_E$ *be a disunification problem,* $V \subseteq Var(\Gamma, \Delta)$.

1. a) A substitution with exceptions $\sigma - \Psi$ *on V solves the disunification problem or is E-inconsistent on V iff*

$$\mathcal{F}_E \models \forall([\sigma] \wedge \neg\exists_{-V}[\Psi] \Rightarrow \Gamma \wedge \Delta)$$

b) A set of consistent substitutions with exceptions $\{ \sigma_\iota - \Psi_\iota : \iota \in I \}$ *on V is a complete representation of the disunification problem iff*

$$\mathcal{F}_E \models \forall_V (\Gamma \wedge \Delta \Leftrightarrow \bigvee_{\iota \in I} (\exists_{-V}[\sigma_\iota] \wedge \neg\exists_{-V}[\Psi_\iota]))$$

2. Corresponding results hold for constrained substitutions, with

$\exists_{-V}[\sigma_\iota] \wedge \neg\exists_{-V}[\Psi_\iota]$ *replaced by* $\exists_{-V}([\sigma] \wedge \neg\exists_{-Var(\sigma V)}[\Theta])$*, etc.*

4.2 Representation of Solutions

The following ***Representation Theorem*** shows that the solutions of a disunification problem can be obtained as instances of certain sets of substitutions with exceptions or constrained substitutions that are generated by solving unification problems only. It depends on some easy transformations of the solution sets (cf. Lassez et al. 1987):

$$S_E(\Gamma, \Delta) = U_E(\Gamma) - \bigcup_{p \neq q \in \Delta} U_E(p = q) = U_E(\Gamma) - \bigcup_{p \neq q \in \Delta} U_E(\Gamma, p = q).$$

Part 1 of the theorem corresponds to the first transformation, part 2 and 3 to the second one. It essentially says that in this transformation the sets of solutions can be replaced by complete solution sets. Notice that when they exist, we can also use minimal unifier sets instead of arbitrary complete ones.

4.13 Representation Theorem: *Let* $\langle \Gamma, \Delta \rangle_E$ *be a disunification problem, let* $V \supseteq Var(\Gamma, \Delta)$*, and let* $cU(\Lambda)$ *denote a complete solution set on* $Var(\Lambda)$ *for the unification problem* $\langle \Lambda \rangle_E$.

1. Let $\Psi := \bigcup_{p \neq q \in \Delta} cU(p = q)$*, then*

$$\{ \sigma - \Psi : \sigma \in cU(\Gamma) \text{ but not } \sigma \geq_E \Psi[V] \}$$

is a complete set of substitutions with exceptions for $\langle \Gamma, \Delta \rangle_E$.

2. Let $\Phi_\sigma := \bigcup_{p \neq q \in \Delta} cU(\sigma p = \sigma q)$*, then*

$$\{ \sigma \| \Phi_\sigma : \sigma \in cU(\Gamma) \text{ but not } \varepsilon \geq_E \Theta_\sigma [Var(\sigma V)] \}$$

is a complete set of constrained substitutions for $\langle \Gamma, \Delta \rangle_E$.

3. Let $\Theta_\sigma := \bigcup_{\psi \in \Psi} cU(\sigma v = \psi v : v \in V)$*, for all* $\sigma \in cU(\Gamma)$*, then*

$$\{ \sigma \| \Theta_\sigma : \sigma \in cU(\Gamma) \text{ but not } \varepsilon \geq_E \Theta_\sigma [Var(\sigma V)] \}$$

is a complete set of constrained substitutions for $\langle \Gamma, \Delta \rangle_E$.

Proof: 1. λ solves $\langle \Gamma, \Delta \rangle_E$ then $\lambda s =_E \lambda t$ for all $s = t \in \Gamma$ and $\lambda p \neq_E \lambda q$ for all disequations $p \neq q \in \Delta$. Hence there is a $\sigma \in cU(\Gamma)$ with $\lambda \geq_E \sigma[V]$. Assume $\lambda \geq_E \Psi [V]$, then there is some $p \neq q \in \Delta$ and some $\psi \in cU(p = q)$ with $\lambda \geq_E \psi[V]$. Hence $\lambda p =_E \lambda q$, a contradiction.

Conversely let $\lambda \geq_E \sigma[V]$ for some $\sigma \in cU(\Gamma)$, but not $\lambda \geq_E \Psi [V]$. Then obviously $\lambda s =_E \lambda t$ for all $s = t \in \Gamma$. Assume $\lambda p =_E \lambda q$ for some $p \neq q \in \Delta$, then λ is an instance of some $\psi \in cU(p = q)$. That is $\lambda \geq_E \Psi [V]$, a contradiction.

2. λ solves $\langle \Gamma, \Delta \rangle_E$ then $\lambda s =_E \lambda t$ for all $s = t \in \Gamma$ and $\lambda p \neq_E \lambda q$ for all $p \neq q \in \Delta$. Hence there is some $\sigma \in cU(\Gamma)$ and some γ with $\lambda =_E \gamma\sigma[V]$. Assume $\gamma \geq_E \Phi_\sigma [Var(\sigma V)]$, then there is some $p \neq q \in \Delta$ and some $\varphi \in cU(\sigma p = \sigma q)$ with $\gamma \geq_E \varphi [Var(\sigma V)]$. Hence $\gamma\sigma p =_E \gamma\sigma q$, and therefore $\lambda p =_E \lambda q$, a contradiction.

Conversely let $\lambda =_E \gamma\sigma [V]$ for some $\sigma \in cU(\Gamma)$ and some γ, such that γ is not an instance of Φ_σ. Then obviously $\lambda s =_E \lambda t$ for all $s = t \in \Gamma$. Assume $\lambda p =_E \lambda q$ for some $p \neq q$ in Δ, that is $\gamma\sigma p =_E \gamma\sigma q$. Hence there is some $\varphi \in cU(\sigma p = \sigma q)$, with $\gamma \geq_E \varphi [Var(\sigma V)]$. Therefore $\gamma \geq_E \Phi_\sigma [Var(\sigma V)]$, again a contradiction.

3. λ solves $\langle \Gamma, \Delta \rangle_E$ then $\lambda s =_E \lambda t$ for all $s = t \in \Gamma$ and $\lambda p \neq_E \lambda q$ for all $p \neq q \in \Delta$. Hence there is some $\sigma \in cU(\Gamma)$ and some γ with $\lambda =_E \gamma\sigma[V]$. Assume $\gamma \geq_E \Theta_\sigma [Var(\sigma V)]$, then there is some $p \neq q \in \Delta$, some $\tau \in cU(p = q)$, and some $\theta \in cU(\sigma V = \tau V)$ with $\gamma \geq_E \theta [Var(\sigma V)]$. Hence we have $\theta\sigma x =_E \theta\tau x$ for all variables $x \in V$ and there is some β with $\gamma y =_E \beta\theta y$ for all $y \in Var(\sigma V)$. Then the following equality chain holds: $\lambda x =_E \gamma\sigma x =_E \beta\theta\sigma x =_E \beta\theta\tau x$ for all $x \in V$. This is $\lambda \geq_E \tau [V]$, and hence $\lambda p =_E \lambda q$, a contradiction.

Conversely let $\lambda =_E \gamma\sigma [V]$ for some $\sigma \in cU(\Gamma)$ and some γ, such that γ is not an instance of Θ_σ. Then obviously $\lambda s =_E \lambda t$ for all $s = t \in \Gamma$. Assume $\lambda p =_E \lambda q$ for some $p \neq q$ in Δ, then λ is an instance of some $\tau \in cU(p = q)$, hence $\lambda =_E \beta\tau [V]$ for some β. By the separation assumptions $DOM\sigma = DOM\tau = V$ and $VCOD\sigma \cap VCOD\tau = \emptyset$, hence β and γ can be chosen, such that $DOM\beta \cap DOM\gamma = \emptyset$ and $DOM\gamma = Var(\sigma V)$. Hence the substitution θ with $\theta x := \beta x$ for $x \in DOM\beta$ and $\theta x := \gamma x$ for $x \in DOM\gamma$ is well-defined and $\lambda =_E \theta\sigma =_E \theta\tau [V]$, i.e. $\theta \in cU(\sigma V = \tau V)$. By definition $\gamma x = \theta x$ for all variables $x \in Var(\sigma V)$, hence $\gamma \geq_E \Theta_\sigma [Var(\sigma V)]$, again a contradiction. ■

Obviously the theorem also holds if we restrict ourselves to ground solutions. The above substitutions with exceptions and constrained substitutions also represent all ground solutions of the given disunification problem; but remember the note after the Inconsistency Lemma.

As an easy corollary of the Representation Theorem we have that the type unitary or finitary for a given theory is the same with respect to both unification and disunification, and the solutions can be represented by substitutions with finitely many exceptions or constraints.

4.14 Corollary: *a) E is unitary with respect to unification iff E is unitary with respect to disunification. In this case the solutions of disunification problems can be represented by one substitution with finitely many exceptions or constraints.*
b) E is finitary with respect to unification iff E is finitary with respect to disunification. In this case the solutions of disunification problems can be represented by a finite set of substitution with finitely many exceptions or constraints.

The above results show that, provided we have a unification algorithm *E-UNIFY(Γ)* for a theory *E*, that always computes a finite, complete set of E-unifiers for a system of equations Γ and an algorithmus *E-CONSISTENT($\sigma-\Psi$)* for E-consistency tests of substitutions with exceptions or constrained substitutions, we also get a disunification algorithm *E-DISUNIFY(Γ,Δ)* for *E*, that computes a finite and complete set of substitutions with (finitely many) exceptions or constraints solving the disunification problem $\langle \Gamma, \Delta \rangle_E$. The consistency test can be done by E-unification. We only give the algorithm for substitutions with exceptions corresponding to part 1 of the Representation Theorem; it can easily be extended for the constrained substitutions of part 2 or 3 of the theorem.

4.15 Algorithm *E-DISUNIFY($\Gamma, p_1 \neq q_1, \ldots, p_n \neq q_n$)*

Input: A system of equations Γ and disequations $p_1 \neq q_1, \ldots, p_n \neq q_n$

```
1. $cU := E\text{-}UNIFY(\Gamma)$,
   $cU_1 := E\text{-}UNIFY(p_1 = q_1), \ldots, cU_n := E\text{-}UNIFY(p_n = q_n)$
2. $\Psi := cU_1 \cup \ldots \cup cU_n$
3. $S := \{\sigma-\Psi\colon \sigma \in cU \text{ and } E\text{-}CONSISTENT(\sigma-\Psi)\}$
```

Output: A finite, complete set *S* of substitutions with (finitely many) exceptions solving the input system or the empty set, if the input system has no solutions.

Notice that the system is not solvable iff the equation part is unsolvable or none of the computed substitutions with exceptions has an instance. If some of the unification problems corresponding to the disequations have no solution, then these disequations are redundant, i.e., the disunification problem has the same solutions, if we drop these disequations. More generally, we have the following result, that follows immediately with the Representation Theorem.

4.16 Redundancy Lemma: *Let $\langle \Gamma, \Delta, p_0 \neq q_0 \rangle_E$ be a disunification problem. Equivalent are:*

(i) $p_0 \neq q_0$ *is redundant*

(ii) $U_E(\Gamma, p_0 = q_0) \subseteq \bigcup_{p \neq q \in \Delta} U_E(\Gamma, p = q)$

(iii) $\tau \geq_E \bigcup_{p \neq q \in \Delta} cU_E(\Gamma, p = q)$ *for each* $\tau \in cU_E(\Gamma, p_0 = q_0)$
($cU_E(\Lambda)$ denote complete subsets of the sets $U_E(\Lambda)$)

We cannot get the stronger result of Proposition 21 of Lassez et al. (1987); this holds only for theories *E* that are unitary with respect to unification:

$p_0 \neq q_0$ *is redundant, iff* $U_E(\Gamma, p_0 = q_0) \subseteq U_E(\Gamma, p = q)$ *for some* $p \neq q \in \Delta$.

Here we still have more, since now the solutions of every unification problem can be represented by a single most general unifier, say σ_0 for $U_E(\Gamma, p_0 = q_0)$ and σ_i for $U_E(\Gamma, p_i = q_i)$ for each $p_i \neq q_i \in \Delta$:

$$p_0 \neq q_0 \textit{ is redundant, iff } \sigma_0 \geq_E \sigma_i \textit{ for some } i.$$

4.3 Propagation and Merging of Solutions

In order to solve systems of equations one usually solves them sequentially that is solving the equations step by step setting in the solutions of former equations into the later ones (*propagation*), or one solves all equations quasi-parallel unifying the solutions (*merging*). It is well-known that these approaches lead to complete sets of solutions for a system, provided the intermediate steps produce complete solution sets (the proofs are very similar to those of the Representation Theorem and can be found for example in Bürckert 1989; see also Ohlbach 1986, Herold 1987). We generalize these results for substitutions with exceptions and constrained substitutions.

4.17 Propagation Lemma: *Let $\langle \Gamma_1, \Delta_1 \rangle_E$ and $\langle \Gamma_2, \Delta_2 \rangle_E$ be two E-disunification problems, $V \supseteq Var(\Gamma_1, \Delta_1, \Gamma_2, \Delta_2)$, and $cU(\Lambda)$ denote a complete solution set on $Var(\Lambda)$ for the unification problem $\langle \Lambda \rangle_E$. Let $\Psi := \bigcup_{p \neq q \in \Delta 1 \cup \Delta 2} cU(p = q)$, then*

$$\{(\tau\sigma)|_V - \Psi\text{: } \sigma \in cU(\Gamma_1),\ \tau \in cU(\sigma\Gamma_2) \text{ but not } \tau\sigma \geq_E \Psi\, [V]\}$$

is a complete solution set for $\langle \Gamma_1, \Gamma_2, \Delta_1, \Delta_2 \rangle_E$.

Proof: It is well-known that $\{(\tau\sigma)|_V\text{: } \sigma \in cU(\Gamma_1),\ \tau \in cU(\sigma\Gamma_2)\}$ is a complete solution set for $\langle \Gamma_1, \Gamma_2 \rangle_E$. Hence the theorem follows with the Representation Theorem. ∎

4.18 Merging Lemma: *Let $\langle \Gamma_1, \Delta_1 \rangle_E$ and $\langle \Gamma_2, \Delta_2 \rangle_E$ be two E-disunification problems, let $V \supseteq Var(\Gamma_1, \Delta_1, \Gamma_2, \Delta_2)$, and let $cU(\Lambda)$ always denote a complete solution set on $Var(\Lambda)$ for the unification problem $\langle \Lambda \rangle_E$. Let $\Theta_{\sigma\tau} := \bigcup\{cU(\sigma v = \psi v,\ \psi v = \tau v\text{: } v \in V)\text{: } \psi \in cU(p = q) \text{ for } p \neq q \in \Delta_1 \cup \Delta_2\}$.
Then:
$\{(\delta\sigma)|_V /\!/ \Theta_{\sigma\tau}\text{: } \sigma \in cU(\Gamma_1), \tau \in cU(\Gamma_2), \delta \in cU(\sigma v = \tau v\text{: } v \in V), \text{but not } \varepsilon \geq_E \Theta_{\sigma\tau}\, [Var(\delta\sigma V)]\}$
is a complete solution set for $\langle \Gamma_1, \Gamma_2, \Delta_1, \Delta_2 \rangle_E$ (notice, that $\delta\sigma =_E \delta\tau\, [V]$).*

Proof: Again the proof follows with the Representation Theorem and the fact that $\{(\delta\sigma)|_V\text{: } \sigma \in cU(\Gamma_1),\ \tau \in cU(\Gamma_2),\ \delta \in cU(\sigma v = \tau v\text{: } v \in V)\}$ is a complete solution set for $\langle \Gamma_1, \Gamma_2 \rangle_E$. ∎

Notice, that by the symmetry of the Merging Theorem we have also shown the solutions are independent of the ordering of equations and disequations in the system.

We can also formulate a propagation theorem for the second form of substitutions with exceptions and a merging theorem for the first form. Notice, that the first and second part of the Representation Theorem are special cases of propagation and merging, where Δ_1 and Γ_2 are empty. Additionally we can have some versions of merging or propagation analoguously to part 3 of the Representation Theorem. However, these cases are similar to the above two versions, but still more technical.

4.4 Restricting Variables

For a given disunification problem $\langle \Gamma, \Delta \rangle_E$, one may not always be concerned about solution values for all the variables in $Var(\Gamma, \Delta)$. Some variables may be auxiliary variables, which only play an existential role. A value for them must exist, but the actual value is of no consequence. For example, all solutions for the problem

$$\langle f(x, g(u)) = f(y, y), x \neq g(a) \rangle_{\emptyset}$$

for the variables in $V = \{x,y,u\}$ can be represented by

$$\{x \leftarrow g(u), y \leftarrow g(u)\} - \{x \leftarrow g(a)\}.$$

Suppose, however, that, only solutions for variables in $V = \{y\}$ are required, so the variables x and u play an existential role. In this case, solutions can be represented by

$$\{y \leftarrow g(u)\} - \{y \leftarrow g(a)\}.$$

The exception needed to be restructured in this case to account for the fact that x no longer existed in the substitution. Denote this process of removing existential parameters as finding the *restriction* of a substitution with exceptions. The treatment below introduces the general problem of existential parameters and then shows how to resolve the problem using restrictions (cf. Nutt et al. 1989, for the case of solving equations only).

Let us denote the problem to resolve a system $\Gamma \cup \Delta$ of equations and disequations over $\mathcal{F}_E$ for just the variables in V as an *E-disunification problem on V*, written $\langle \Gamma, \Delta \rangle_E$ *on V*. Variables in $Var(\Gamma,\Delta) - V$ are the *existential parameters* for the problem.

An *E-solution* of $\langle \Gamma, \Delta \rangle_E$ on V is every substitution $\sigma \in \mathcal{SUB}(V, \mathcal{V})$, such that there exists a substitution $\tau \in \mathcal{SUB}(Var(\Gamma,\Delta) - V, \mathcal{V})$ with $\sigma\tau s =_E \sigma\tau t$ for each $s = t \in \Gamma$ and $\sigma\tau p \neq_E \sigma\tau q$ for each $p = q \in \Delta$. Whether a set of substitutions with exceptions or constrained substitutions is a correct and complete representation of a disunification problem on V is defined as before.

In accord with Lemma 4.2, the logical interpretation of a solution to $\langle \Gamma, \Delta \rangle_E$ on V is an assignment to variables in V such that $\exists_{-V} (\Gamma \wedge \Delta)$ is true in $\mathcal{F}_E$.

4.19 Corollary to Lemma 4.2: *1 a). A substitution with exceptions $\sigma - \Psi$ on V solves the disunification problem $\langle \Gamma, \Delta \rangle_E$ on V or is E-inconsistent on V*

$$\textit{iff } \mathcal{F}_E \models \forall([\sigma] \wedge \neg\exists_{-V}[\Psi] \Rightarrow \exists_{-V} (\Gamma \wedge \Delta))$$

1b) and 2. Corresponding results hold for completeness and for constrained substitutions.

To find a solution for a disunification problem $\langle \Gamma, \Delta \rangle_E$ on V, it is clearly sufficient to find a solution to the disunification problem $\langle \Gamma, \Delta \rangle_E$ (on $Var(\Gamma, \Delta)$) and then remove the existential parameters, that is, the variables in $Var(\Gamma, \Delta) - V$.

4.20 Definition: The *E-restriction* of a substitution with exceptions $\sigma - \Psi$ on U to a set of variables $V \subseteq U$ is a substitution with exceptions $\delta - \Lambda$ on V such that $\lambda \geq_E \sigma - \Psi\,[U]$ implies $\lambda/_V \geq_E \delta - \Lambda\,[V]$, and $\lambda \geq_E \delta - \Lambda\,[V]$ implies there exists a substitution τ such that $\lambda\tau \geq_E \sigma - \Psi\,[U]$. This *E-restriction* is denoted $\sigma - \Psi/_V$, where E and U are to be taken from context. ■

A corresponding definition naturally holds for the E-restriction of a constrained substitution, and for restrictions in algebras other than the free algebra. The definition is also consistent with the common one for substitutions. An algorithm for constructing an E-restriction is not nearly as simple as it is for substitutions, however. An algorithm makes use of the following other algorithms. The algorithm *E-TRANSLATE*$(\sigma-\Psi, U)$ computes a constrained substitution equivalent to the substitution with exceptions $\sigma-\Psi$ on U. The Translation Lemma part 1 specifies such an algorithm for finitary equality theories. The algorithm *E-TRANSLATE*$^{-1}(\theta \| \Theta, V)$ computes the corresponding inverse translation, for example, via Translation Lemma part 2. Finally, the algorithm *E-UNIV-QUANT*(Θ, U, V) computes a finite set of substitutions Ω such that

$$\mathcal{F}_E \models \forall_V (\exists_{-V}[\Omega] \Leftrightarrow \forall_{U \setminus V} \exists_{-U}[\Theta])).$$

Some *UNIV-QUANT* algorithms are given below.

4.21 Algorithm *E-RESTRICT*$(\sigma-\Psi, U, V)$

Input: A substitution with exceptions $\sigma-\Psi$ on U and a set $V \subseteq U$
(It is assumed all existing substitutions are restricted to U and contain no variables in common other than those in U.)

```
1. $\theta \| \Theta$ := E-TRANSLATE( $\sigma-\Psi$, U)
2. $\Theta'$ := E-UNIV-QUANT($\Theta$, Var($\theta U$), Var($\theta V$))
3. $\delta-\Lambda$ := E-TRANSLATE$^{-1}$($\theta/_V \| \Theta'$, V)
```

Output: A substitution with exceptions $\delta-\Lambda$ on V that is the E-restriction of $\sigma-\Psi$ on U to V.

4.22 Restriction Lemma: *The E-restriction algorithm E-RESTRICT is correct.*

Proof: The algorithm corresponds to the following transformations:

$$\begin{aligned} & \exists_{-V} (\exists_{-U} [\sigma] \wedge \neg\exists_{-U}[\Psi]) \\ \Leftrightarrow\ & \exists_{-V} \exists_{-U} ([\theta] \wedge \neg\exists_{-Var(\theta U)}[\Theta]) && \text{(by step 1)} \\ \Leftrightarrow\ & \exists_{-V} \exists_{-U} ([\theta/_V] \wedge \neg\forall_{Var(\theta U) \setminus Var(\theta V)}\exists_{-Var(\theta U)}[\Theta]) \\ \Leftrightarrow\ & \exists_{-V} \exists_{-U} ([\theta/_V] \wedge \neg\exists_{-Var(\theta V)}[\Theta']) && \text{(by step 2)} \\ \Leftrightarrow\ & \exists_{-V} [\delta] \wedge \neg\exists_{-V}[\Lambda] && \text{(by step 3)} \end{aligned}$$ ■

Notice that to construct the E-restriction of a constrained substitution, only step 2 in the algorithm is required. The relationship between E-restriction and translating constrained substitutions to substitutions with exceptions is a strong one. We also have:

$$\textit{E-TRANSLATE}^{-1}(\theta \| \Theta, V) \ = \ \textit{E-RESTRICT}(\theta-\Theta, V \cup Var(\theta), V).$$

Finally, the definitions for restriction and the above results easily transfer to algebra or model families other than the free algebra. For the free algebra of syntactic equality, a *UNIV-QUANT* algorithm is particularly simple.

4.22 Lemma: *For the free algebra of syntactic equality,* $\mathcal{F}_\emptyset$,
$\emptyset$-*UNIV-QUANT*$(\Theta, U, V) := \{\theta \in \Theta : (U \setminus V) \cap Var(\theta) = \emptyset\}$.

Proof: For any assignment to the variables in V, there always exist distinct constants that can be assigned to the variables in $U \setminus V$ forcing every $[\theta]$ such that $(U \setminus V) \cap Var(\theta) \neq \emptyset$ to be false. The remaining substitutions are not effected by the quantification. ■

Clearly, an identical result follows for A. Colmerauer's domain of infinite trees (1984) when an infinite supply of function symbols exists, and for theories such as *AC*.

5. Applications

There are several applications for E-disunification. In recent papers logic programming (Goguen & Meseguer 1986, Jaffar et al. 1986, Gallier & Raatz 1986, Bürckert 1986) as well as term rewriting (Lankford & Ballantyne 1977, Peterson & Stickel 1981, Stickel 1984, Jouannaud & Kirchner 1984) have been extended for unification or matching with respect to equational theories. The approach of A. Colmerauer - disunification as a constraint solving process for logic programming - is generalized in a naturally way to the above extension (section 5.1). Also H. Comon's application for disunification to show sufficient completeness for algebraic specifications given by term rewriting systems might be extended in such a way (equational theories may be used to specify non-free datatypes), cf. (Comon & Lescanne 1988). The huge number of solutions for AC-unification problems (Bürckert et al. 1989$_2$), where AC is the theory of associative and commutative functions, can be reduced by representing these solutions with substitutions with exceptions that solve suitable AC1-disunification problems (section 5.2). Finally we consider certain applications in resolution based theorem proving, where disunification might be used to avoid certain redundancies in the search space (section 5.3).

5.1 Logic Programming and Disunification

Negation has been a continuing problem in logic programming since its inception. A limited approach is negation as failure (Clark 1978, Lloyd 1984). But as illustrated in the introduction, this approach will always remain limited because some queries do not have solutions that can be expressed as substitutions. We outline a more general approach here.

We shall take an abstract view of logic programming. A logic programming system transforms a query Q represented as a logical formula into an intermediate solution form S and then simplifies that intermediate form into an equivalent solution form A. The transformation process makes use of the *logic program* available to the system and is such that S implies Q in some special model or structure consistent with the logic program. The simplification process preserves equivalence in the model or structure.

A simple example demonstrates this position well. Pure Prolog is the use of definite clause logic programs on the empty equality theory evaluated, for instance, using breadth-first linear resolution.

logic program:	definite clauses
transformation process:	the application of *modus ponens*
intermediate solution forms:	existentially quantified conjunctions of equations
simplification process:	the unification algorithm
solution forms:	substitutions
model space:	Herbrand models

The various froms of linear resolution efficiently splice the transformation and simplification processes.

In this subsection we present an abstract system for logic programming that uses the full power of disequation processing developed in Section 4.

First some definitions. An *equational formula* is a formula of first order logic such that the only predicate symbol contained is the equality symbol. These formulae will be our intermediate solution form. The logic programs we will consider are completed general logic programs (Lloyd 1984). These can be generated, for instance, from extended programs constructed from arbitrary first order formulae (Lloyd and Topor, 1984). They are defined as follows. A *general program clause* is of the form $A \Leftarrow W$ where W is a conjunction of literals (atoms or negated atoms). A *general program* is a set of such clauses. A *completed definition* for an n-ary predicate p from a general program P is a formula of the form

$$\forall (\, p(x_1,\ldots,x_n) \Leftrightarrow E_1 \vee \ldots \vee E_m \,)$$

where each E_i is of the form $\exists_{-X}(\, x_1 = t_{1i} \wedge \ldots \wedge x_n = t_{ni} \wedge W_i \,)$, given that there are exactly m general program clauses in the general program P of the form $p(t_{1i},\ldots,t_{ni}) \Leftarrow W_i$, and $X=\{x_1,\ldots,x_n\}$ are distinct new variables. A *completed general program* is the set of distinct completed definitions from a general program.

One form of computation for completed general programs is *SLDNF-refutation* (Lloyd 1984, see also Clark 1978). We will not elaborate here on the details of this refutation strategy, suffice it say the strategy has a severe restriction in that negative literals can only be evaluated *if they are ground.* The restriction is necessary to ensure the soundness of the strategy, and can be seen to be a direct consequence of the fact that the only intermediate solution forms allowed in such strategies are substitutions. Non-ground negative literals may well require disequations to express their full range of solutions.

In Section 4 we put all the necessary machinery in place so that any equational formula can be simplified to one of our solution forms, a set of substitutions with exceptions. We formalise this in Lemma 5.1 for the particular context of a free algebra, although the approach generalises to other algebras, for instance the initial algebra, if the necessary basic algorithms exist. Lemma 5.1 means that a whole new class of computation strategies are possible, for instance, not suffering the restrictions of SLDNF-refutation.

5.1 Lemma:*Given an equational or definite clause equality theory E that is finitary, and algorithms E-UNIFY, E-CONSISTENT, E-TRANSLATE^{-1} and E-UNIV-QUANT as specified in Sections 4.3 and 4.4, and an equational formula F with free variables V. A finite reduced set of substitutions with exceptions $\{\sigma_\iota - \Psi_\iota : \iota \in I\}$ on V can be constructed that is equivalent to F in the free algebra $\mathcal{F}_E$. That is*

$$\mathcal{F}_E \models \forall_V (\, F \;\Leftrightarrow\; \vee_{\iota \in I} (\, \exists_{-V}[\sigma_\iota] \wedge \neg\exists_{-V}[\Psi_\iota] \,)) \,.$$

Proof: Clearly, the space of finite sets of substitutions with exceptions is closed under the operations of negation and disjunction. The logical form of a set of substitutions with exceptions demonstrates this well. Due to the Restriction Lemma, we also have in the situation we are considering that a set of substitutions with exceptions can be existentially quantified. Finally, unquantified conjunctions of equations and disequations can be transformed into a set of substitutions with exceptions by the Representation Theorem. It follows by induction on the structure of F that from F we can construct an equivalent set of substitutions with exceptions $\{\sigma_\iota - \Psi_\iota : \iota \in I\}$ on V. The Instantiation Lemma and its various corollaries show that this can be subsequently reduced if we have an instantiation test available. Notice that in the free algebra, whether a substitution is an instance of a set of substitutions (in the sense of the definition in Section 4.1) can be tested by a unification algorithm. This follows from Lemma 3.4. In general, however, the *E-CONSISTENT* algorithm could perform such a test, by the Inconsistency Lemma. ■

Consider the following non-deterministic, possibly non-terminating transformation process.

```
Input:  A query $Q$ that is a literal, and a completed general logic program $P$.
  1. Let $F := Q$.
     2. Repeatedly perform one of the following:
        2a. If an atom $A = p(s_1,\ldots,s_n)$ occurs inside an even number of
            negations in $F$,
            replace $A$ in $F$ by $\exists_{-Var(A)} (s_1 = t_{1,i} \wedge \ldots \wedge s_n = t_{n,i} \wedge W_i)$ for some $i$,
            where the predicate $p$ is as given in the definition of a completed
            definition.
        2b. If an atom $A = p(s_1,\ldots,s_n)$ occurs inside an odd number of
            negations in $F$,
            replace $A$ in $F$ by $\exists_{-Var(A)} (s_1 = x_1 \wedge \ldots \wedge s_n = x_n \wedge (E_1 \vee \ldots \vee E_m))$,
            where the predicate $p$ is as given in the definition of a completed
            definition.
        2c. If an atom occurs inside an odd number of negations in $F$,
            replace the atom in $F$ by *true*.
     Until $F$ is an equational formula.
Ouput:  An equational formula $F$ such that $P \models \forall( Q \Leftarrow F )$.
```

As an example, the predicate *member(x,y)* corresponds to $x \in y$, and the predicate *set-difference(x,y,z)* corresponds to $z = x \setminus y$. These predicates can be defined logically as follows. (The "." notation for lists is used, for instance, *[1,2,3]* corresponds to *1.2.3.[].*)

$$member(x,y) \Leftrightarrow \exists_u (y = x.u) \vee \exists_{u,v} (y = u,v \wedge member(x,v))$$

$$set\text{-}difference(x,y,z) \Leftrightarrow \forall_u (member(u,z) \Leftrightarrow member(u,x) \wedge \neg member(u,y))$$

Using Lloyd and Topor's method for transforming first order formulae to general programs (1984), these give the following general program, to be evaluated on syntactic equality.

$member(x,x.u).$

$member(x,u.v) \Leftarrow member(x,v).$

$set\text{-}difference(x,y,z) \Leftarrow \neg sd1(x,y,z).$

$sd1(x,y,z) \Leftarrow member(u,x) \wedge \neg member(u,y) \wedge \neg member(u,z).$

$sd1(x,y,z) \Leftarrow \neg member(u,x) \wedge member(u,z).$

$sd1(x,y,z) \Leftarrow member(u,y) \wedge member(u,z).$

Suppose we wish to answer the query *set-difference([1,2,X], [1,3],Y)*? The transformation process first yields (some forms have been simplified to ease presentation)

$$\neg\exists_u (\mathit{member}(u, [1,2,X]) \wedge \neg \mathit{member}(u, [1,3]) \wedge \neg \mathit{member}(u,Y))$$
$$\wedge \neg\exists_u (\neg \mathit{member}(u, [1,2,X]) \wedge \mathit{member}(u,Y))$$
$$\wedge \neg\exists_u (\mathit{member}(u, [1,3]) \wedge \mathit{member}(u,Y)) ,$$

then applying the process to transform *member* yields, for one set of choices,

$$\neg\exists_u ((u=1 \vee u=2 \vee u=X) \wedge \neg u=1 \wedge \neg \exists_v Y = u.v)$$
$$\wedge \neg\exists_u (\neg u=2 \wedge (\exists_v Y = u.v \vee \exists_{v.w} Y = v.u.w \vee \exists_{v.w,r} Y = v.r.w))$$
$$\wedge \neg\exists_u ((u=1 \vee u=3) \wedge (\exists_v Y = u.v \vee \exists_{v.w} Y = v.u.w \vee \exists_{v.w,r} Y = v.r.w)) .$$

Simplifying each line separately in the free algebra for syntactic equality yields

$$\exists_v Y = 2.v \wedge (X=1 \vee \exists_v Y =X.v)$$
$$\wedge (\neg \exists_{u,v} Y = u.v \vee \exists_v Y = 2.v) \wedge \neg \exists_{v,r,w} Y = v.r.w$$
$$\wedge \neg\exists_v Y = 1.v \wedge \neg\exists_v Y = 3.v \wedge \neg \exists_{v.w,r} Y = v.r.w .$$

This then simplifies to two solutions

$$\{ Y \leftarrow 2.v, X \leftarrow 1 \} - \{ Y \leftarrow v.r.w \} ,$$
$$\{ Y \leftarrow 2.v, X \leftarrow 2 \} - \{ Y \leftarrow v.r.w \} .$$

The remaining solutions can be found by making other choices when transforming *member*.

Our abstract system then is as follows:

logic program:	a completed general program
transformation process:	as given above
intermediate solution forms:	equational formula
simplification process:	disunification, restriction, reduction as inSection 4
solution forms:	sets of substitutions with exceptions
model space:	the free algebra for some equality theory

To make this abstract system practical, the following problem remains: How can the simplification and transformation processes be spliced to yield an efficient algorithm, as linear resolution does for the case of pure Prolog. Perhaps a generalisation of negation as failure is possible that incorporates disequations as well.

5.2 AC-Unification is AC1-Disunification

As mentioned in the introduction disunification can be used to prevent extensive splitting of solutions. Instead of solving equations in the theory *AC* of an associative and commutative function, we can solve them in the theory *AC1*, that is AC with some unit *1*, but with the constraints that the variables must not become equal to *1*, that is, we solve them together with the disequations $x \neq 1$ for all the variables. When we avoid evaluation of the constraints $x \neq 1$ in these AC1-disunification problems as long as possible, we can avoid an exponential grow up of the number of solutions. The idea is to propagate these constraints for example in theorem proving applications of E-unification (Plotkin 1972) into resolvents that are created with the substitutions solving the equation part only, but only when these substitutions are not instances of the collected exceptions (i.e. the substitution with collected(!) exceptions are still AC1-consistent). A further advantage of this method is that we generate in some sense multi-resolvents, since the substitutions with exception may represent more than one most general AC-unifier.

Let AC be an equational theory over a signature Σ consisting of free constants, free function symbols, and a binary infix function ". " being associative and commutative, that is we have the axiomatization

$$AC := \{(x \,.\, (y \,.\, z)) = ((x \,.\, y) \,.\, z),\ x \,.\, y = y \,.\, x\}.$$

Moreover let Σ_1 be Σ extended with a constant 1 being a unit with respect to the AC-function. Hence we have the further equational theory $AC1$ of free Abelian monoids with free functions with the axiomatization

$$AC1 := \{(x \,.\, (y \,.\, z)) = ((x \,.\, y) \,.\, z),\ x \,.\, y = y \,.\, x,\ 1 \,.\, x = x\}.$$

We use a normalized representation of AC-terms and AC1-terms by ordered strings with exponentiation (dropping all occurrences of the unit):

Both the AC-term	$((a \,.\, x) \,.\, (f(a \,.\, x, a \,.\, (b \,.\, a)) \,.\, (c \,.\, x)))$
and the AC1-term	$(1 \,.\, ((a \,.\, x) \,.\, (f((a \,.\, x), ((a \,.\, (b \,.\, a)) \,.\, 1)) \,.\, (c \,.\, x))))$
are represented by	$acf(ax, a^2b)x^2$.

Now, we are interested in solving equations under both the theory $AC1$ and the theory AC. Both kinds of unification problems can be solved by computing the minimal non-negative integer solutions of suitable corresponding linear Diophantine equations (Livesey & Siekmann 1976, Stickel 1976, 1987, Hullot 1980, Fages 1985, Fortenbacher 1985, Kirchner 1985, Büttner 1986, Herold & Siekmann 1986, Herold 1986). In the case of AC1-unification there is a one-to-one correspondence between the Diophantine solutions and the most general AC1-unifiers, while in the case of AC-unification we must in addition instantiate all subsets of the variables of these most general AC1-solutions with the unit 1 to obtain the most general AC-unifiers. By this post-process in general the number of AC-solutions is growing exponentially in the number of variables introduced by the AC1-solutions:

> The problem $\langle xyz = v^4 \rangle_{AC1}$ has one most general solution introducing 15 new variables, while the corresponding AC-problem $\langle xyz = v^4 \rangle_{AC}$ has about 2^{15} (i.e. more than 32 000) most general AC-unifiers (Bürckert et al. 19892).

Hence it is convenient to avoid an explicit generation of all these most general AC-unifiers by representing them by AC1-unifiers. How this representation works is stated by the following theorem on the relationship between AC- and AC1-unification (Livesey & Siekmann 1976, Herold & Siekmann 1986, Herold 1987).

5.2 Theorem: *1.* $\mathcal{F}_{AC} \models \exists.\Gamma$ *iff* $\mathcal{F}_{AC1} \models \exists.\Gamma \wedge \Delta$ *with* $\Delta = \{x \neq 1 : x \in Var(\Gamma)\}$.

2. A substitution σ solves the AC-unification problem $\langle \Gamma \rangle_{AC}$ iff σ solves the AC1-problem $\langle \Gamma \rangle_{AC1}$ and $\sigma x \neq_{AC1} 1$ for all $x \in Var(\Gamma)$.

3. Let μU_{AC1} be a minimal solution set for $\langle \Gamma \rangle_{AC1}$. Then the set

$$\mu U_{AC} := \{(\nu\sigma)|_V : \nu \in N_\sigma, \sigma \in \mu U_{AC1}, \nu\sigma x \neq 1\ \forall x \in Var(\Gamma)\}$$

is a minimal solution set for $\langle \Gamma \rangle_{AC}$ (with $N_\sigma := \{\{w \leftarrow 1 : w \in W\} : W \subseteq VCOD\sigma\}$).

Hence by the Representation Theorem the AC-unification problem $\langle \Gamma \rangle_{AC}$ can equivalently be considered as an AC1-disunification problem

$$\langle \Gamma, x \neq 1 : x \in V \rangle_{AC1}$$

with $V = Var(\Gamma)$. For substitutions σ in normalized representation holds:

$$\delta \text{ solves } \langle \Gamma \rangle_{AC} \text{ iff } \delta \geq_{AC1} \sigma\text{–}\Psi\, [V] \text{ for some } \sigma\text{–}\Psi \in U_{AC1}(\Gamma, x \neq 1 : x \in V).$$

(Notice, that every solution of $\langle \Gamma, x \neq 1 : x \in V \rangle_{ACI}$ has a normalized representation without any occurrence of the unit.)

5.3 Disunification for Resolution Based Theorem Provers

There are also some applications for resolution based theorem proving (Chang & Lee 1973, Wos et al. 1984). Here disunification can be used to avoid certain redundancies in proof searching. Consider for example a clause $\{P(x), \neg P(y), Q(x,y)\}$. Then any resolution step with the third literal $Q(x,y)$ of this clause that identifies the arguments of Q will lead to a resolvent that is a tautology, and hence this is an unnecessary step. It is enough to look for resolution candidates that do not identify these arguments or in other words unification of $Q(x,y)$ with some literal $\neg Q(s,t)$ of another clause can be done under the constraint $x \neq y$, that is, we have the disunification problems $\langle x = s, y = t, x \neq y \rangle_\emptyset$. For theorem proving systems with built-in E-unification procedures this will result in E-disunification problems. Obviously one is not interested in substitutions as solutions for these problems, but in substitutions with exceptions or constrained substitutions for easy generation of instances of the substitutions that fulfill the constraints. Similar constraints can be formulated for other redundancy tests in theorem proving procedures, as for example the subsumption or the purity tests.

A rather similar (but only theoretical) application can be found in unification theory itself, namely for certain combination procedures for unification under a combination of equational theories. Some of them use the constant abstraction method for subterms that have syntactically another theory than the top theory of the unification problem (Herold 1987, Schmidt-Schauß 1989). These alien subterms are replaced by new free constants, then the unification algorithm for pure terms of the top theory can be applied, and the abstraction constants have again to be replaced by the corresponding subterms, whereby some post-unification has to be done, since the subterms also might contain variables that are already used in the pure unification step. But since in further steps certain identifications of these abstracted subterms are necessary to retain completeness, one can do these post-unification steps under the constraints that the subterms need not to be identified; we again have E-disunification problems.

6. Conclusion

Since the number of exceptions may be infinite and hence in general it cannot be tested whether such a substitution with exceptions is consistent, the question will arise, whether E-disunification can become undecidable, although E-unification is decidable. The following reduction, due to M. Schmidt-Schauß, shows that this may happen. Let

$$\begin{aligned} DA = \{\, & f(x, f(y, z)) = f(f(x, y), z), \\ & f(g(x, y)\, , z) = g(f(x, z), f(y, z)), \\ & f(x, g(y, z)) = g(f(x, y), f(x, z))\} \end{aligned}$$

be the theory of distributivity and associativity of two binary functions f, g and some free constants, where unification is known to be undecidable (Szabo 1982, Siekmann & Szabo 1986). If we add a constant 0 with the axioms

$$f(x, 0) = f(0, x) = 0,$$
$$g(x, 0) = g(0, x) = 0,$$

then unification in the resulting theory *DA0* is trivially decidable (two terms become equal by substituting each variable with *0*). However, disunification is undecidable, since the DA0-disunification problems $\langle \Gamma, x \neq 0 : x \in Var(\Gamma) \rangle_{DA0}$, where no term of Γ contains the constant *0*, are equivalent to the DA-unification problems $\langle \Gamma \rangle_{DA}$.

Some more results on decidability of unification and disunification can be found in (Bürckert & Schmidt-Schauß 1989).

Here are some open problems to be considered in the future:

Can the substitutions with exceptions be resolved to substitutions, if we are interested in ground solutions only, that is, are there complete sets of substitutions representing all ground solutions of a given E-disunification problem?

This problem is only solved for the empty theory.

Are there for given theories decision procedures for testing "ground consistency" of a substitution with exception, i.e., whether it has still some ground instances?

These questions are useful only in the case of finite complete representation sets, for example to prove sufficient completeness of algebraic specifications with non-free datatypes specified by *E*.

Can the extended completion procedures for term rewriting systems modulo equational theories be adapted to use disunification to avoid explosion of candidates for critical pairs?

This should especially be investigated for the case of term rewriting systems modulo AC.

How can computation strategies such as negation as failure be generalized to incorporate disequations, to complete the abstract approach outlined in section 5.1?

It is interesting that our generalized notion of solutions is still not expressive enough to represent a solution to the simple junior high school mathematics question "what is the square root of the square of x?" (this is of course "x, if $x \geq 0$"). A possible extension of expressiveness in this direction is of course the use of still more general formulae as answers, but how far can/need we go?

Acknowledgements. We would like to gratefully acknowledge a lot of very useful discussions with W. Nutt, M. Schmidt-Schauß, and G. Smolka on the topics of this paper, especially in the model theoretic foundations of solving equations and disequations. These and also some discussions with H. Comon helped in correcting some mistakes in a former version of this paper. N. Eisinger and J. Siekmann carefully read some former drafts of the paper. Any remaining flaws are, of course, the authors' responsibility.

We also want to emphasize once more the contribution of M. Schmidt-Schauß with the above idea of the reduction of undecidability of disunification.

H.-J. Bürckert's research was partially supported by the German Bundesministerium für Forschung und Technologie and the Nixdorf Computer AG under contract of the Joint Research Project ITR8501A. W.L. Buntine's research was supported by an Australian Commonwealth Postgraduate Research Award.

References

Baader, F.: *The Theory of Idempotent Semigroups is of Unification Type Zero.* J. of Autom. Reasoning 2, 1986, p. 283-286.

Birkhoff, G.: *On the Structure of Abstract Algebra.* Proc. Cambridge Phil Soc. 31, 1935, p. 433-454.

Buntine, W.: *Generalized Subsumption and its Applications to Induction and Redundancy.* Artificial Intelligence 36, 1988, to appear.

Buntine, W.: *A Theory of Equations, Inequations, and Solutions for Logic Programming.* Unpublished manuscript, New South Wales Institute of Technology, 1986.

Bürckert, H.-J.: *Lazy Theory Unification in PROLOG: An Extension of the Warren Abstract Machine.* Proc. of 10th German Workshop on Art. Intelligence, Springer, 1986, p. 277-288.

Bürckert, H.-J.: *Matching - A Special Case of Unification?* To appear in J. of Symb. Comp., Special Issue on Unification (ed. C. Kirchner), 1989.

Bürckert, H.-J., Herold, A. & Schmidt-Schauß, M.: *On Equational Theories, Unification, and Decidability.* Proc. of 2nd Conf. on Rewriting Techniques and Applications, Springer, LNCS 256, 1987, p. 204-215; to appear in J. of Symb. Comp., Special Issue on Unification (ed. C. Kirchner), 1989.

Bürckert, H.-J., Herold, A., Kapur, D., Siekmann, J.H., Stickel, M.E., Tepp, M., Zhang, H.: *Opening the AC-Unification Race.* To appear in J. of Autom. Reasoning, 1989.

Bürckert, H.-J. & Schmidt-Schauß, M.: *Some Solvability Results for Equational Problems.* In preparation, 1989.

Burris, S. & Sankappanavar, H.P.: *A Course in Universal Algebra.* Springer, 1979.

Büttner, W.: *Unification in the Datastructure Multisets.* J. of Automated Reasoning, Vol. 2, No. 1, 1986, p. 75-88.

Chang, C.-L. & Lee, R.C.-T.: *Symbolic Logic and Theorem Proving.* Academic Press, 1973.

Clark, K.L.: *Negation as Failure.* In: *Logic and Databases* (eds. H. Gallaire & J. Minker), Plenum Press, 1978, p. 293-322.

Colmerauer, A.: *Equations and Inequations on Finite and Infinite Trees.* Proc. of Intern. Conf. on Fifth Generation Computer Systems, ICOT, 1984, p. 85-99.

Comon, H.: *Sufficient Completeness, Term Rewriting Systems, and Anti-unification.* Proc. of Intern. Conf. on Automated Deduction, Springer LNCS 230, 1986, p. 128-140.

Comon, H.: *Private Communications.* 1987.

Comon, H.: *Unification et Disunification. Théorie et Applications.* Thesis (in French), Université de Grenoble, 1988.

Comon, H. & Lescanne, P.: *Equational Problems and Disunification.* Draft, Université de Grenoble, and Centre de Recherche en Informatique de Nancy CNRS-INRIA, 1987.

Fages, F.: *Formes Canoniques dans les Algèbres Booléennes, et Application à la Démonstration Automatique en Logique de Premier Ordre.* Thèse de 3éme Cycle (in French), Université Paris VI, 1983.

Fages, F.: *Associative-Commutative Unification.* Proc. of 7th Conf. on Automated Deduction, Springer, LNCS 170, 1984, p. 194-208; see also: Technical Report, INRIA, 1985.

Fages, F. & Huet, G.: *Complete Sets of Unifiers and Matchers in Equational Theories.* Proc. of CAAP'83, Springer, LNCS 159, 1983, p. 205-220; see also J. of Theoret. Comp. Sci. 43, 1986, p. 189-200.

Fortenbacher, A.: *Algebraische Unifikation.* Diplomarbeit (in German), Universität Karlsruhe, 1983.

Fortenbacher, A.: *An Algebraic Approach to Unification under Associativity and Commutativity.* Proc. of Conf. on Rewriting Techniques and Applications, Springer, LNCS 202, 1985, p. 381-397.

Gallier, J.H.: *Logic for Computer Science: Foundations of Automated Theorem Proving.* Harper and Row, 1986.

Gallier, J.H. & Raatz, S.: *SLD-Resolution Methods for Horn Clauses with Equality Based on E-Unification.* Proc. of Int. Symp. on Logic Programming, 1986

Goguen, J.A. & Meseguer, J.: *Equality, Types, Modules, and Generics for Logic Programming.* Proc. of 2nd Intern Logic Programming Conf., Uppsala, 1984, p. 115-125.

Goguen, J.A. & Meseguer, J.: *EQLOG - Equality, Types, and Generic Modules for Logic Programming.* In: *Logic Programming: Functions, Relations, and Equations.* Prentice Hall, 1986, p. 295-363.

Grätzer, G.: *Universal Algebra.* Springer, 1979.

Herold, A.: *Combination of Unification Algorithms.* Proc. of 8th Conf. on Automated Deduction, Springer, LNCS 230, 1986, p. 450-469. Also appeared as MEMO-SEKI, Universität Kaiserslautern, 1985.

Herold, A.: *Combination of Unification Algorithms in Equational Theories.* Dissertation, Universität Kaiserslautern, 1987.

Herold, A. & Siekmann, J.H.: *Unification in Abelian Semigroups.* MEMO-SEKI, Universität Kaiserslautern, 1986

Huet, G.: *An Algorithm to Generate the Basis of Solutions to Homogeneous Linear Diophantine Equations.* Information Processing Letters, Vol. 7, No. 3, 1978, p. 144-147.

Huet, G. & Oppen, D.C.: *Equations and Rewrite Rules: A Survey.* In: *Formal Languages: Perspectives and Open Problems.*(ed. R. Book), Academic Press, 1980.

Hullot, J.M.: *Compilation des Formes Canoniques dans des Théories Equationelles.* Thèse du 3ème Cycle (in French), Université de Paris-Sud, 1980.

Jaffar, J., Lassez, J.-L. & Maher, M.: *A Theory of Complete Logic programming with Equality.* Proc. of Conf. on Fifth Generation Computing Systems, ICOT, 1984, p. 175-184.

Jaffar, J., Lassez, J.-L. & Maher, M.: *Logic Programming Language Scheme.* In: *Logic Programming: Functions, Relations, Equations.* (eds. D. DeGroot & G. Lindstrom), Prentice Hall, 1986.

Jaffar, J. & Stuckey, P.J.: *Logic Programming Semantics for Programming with Equations.* Proc. of 3rd Intern. Logic Programming Conf., Springer LNCS 225, 1986, p. 313-326.

Jouannaud, J.P. & Kirchner, H.: *Completion of a Set of Rules Modulo a Set of Equations.* Proc. of 11th ACM Conf. on Principles of Programming Languages, 1984.

Kapur, D. & Narendran, P.: *An Equational Approach to Theorem Proving in First-Order Predicate Calculus.* Research Report, General Electric, 1985.

Kirchner, C.: *Methodes et Outils de Conception Systematique d'Algorithmes d'Unification dans les Théories Equationelles.* Thèse de Doctorat d'Etat (in French), Université de Nancy, 1985.

Kirchner, C.: *Special Issue on Unification.* J of Symb. Comp., 1989, to appear.

Kirchner, C. & Kirchner, H.: *Implementation of a General Completion Procedure Parametrized by Built-in Theories and Strategies.* Proc. of the EUROCAL Conference, 1985.

Lankford, D. & Ballantyne, R.M.: *Decision Procedures for Simple Equational Theories with Commutative-Associative Axioms: Complete Sets of Commutative-Associative Reductions.* Internal Report, University of Texas, Austin, 1977.

Lassez, J.-L., Maher, M.J. & Marriot, K.: *Unification Revisited.* In Minker, J. (ed.): *Foundations of Deductive Databases nand Logic Programming.* Morgan-Kaufmann, 1987.

Livesey, M. & Siekmann, J.H.: *Unification of AC-Terms (Bags) and ACI-Terms (Sets).* Internal Report, University of Essex, 1975, and Universität Karlsruhe, 1976.

Lloyd, J.W.: *Foundations of Logic Programming.* Springer, 1984.

Lloyd, J.W. & Topor, R.: *Making Prolog more Expressive.* Technical Report 84-8, University of Melbourne, 1984.

Nutt, W., Rety, P. & Smolka, G.: *Basic Narrowing Revisited.* To appear in J. of Symb. Comp., Special Issue on Unification (ed. C. Kirchner), 1989.

Ohlbach, H.J.: *Link Inheritance in Abstract Clause Graphs.* J. of Automated Reasoning, Vol 3, No 1, 1987, p. 1-34.

Peterson, G.E. & Stickel, M.E.: *Complete Sets of Reductions for Equational Theories with Complete Unification Algorithms.* JACM, Vol 28, No 2, 1981, p. 322-364.

Plotkin, G.: *Building in Equational Theories.* Machine Intelligence 7, 1972, p. 73-90.J.A. Robinson: *A Machine Oriented Logic Based on the Resolution Principle.* JACM, Vol 12, No 1, 1965, p. 23-41.

Schmidt-Schauß, M.: *Unification under Associativity and Idempotence is of Type Nullary.* J. of Autom. Reasoning 2, 1986, p. 277-282.

Schmidt-Schauß, M.: *Combination of Arbitrary Equational Theories With Simple Equational Theories.* SEKI-Report, Universität Kaiserslautern, 1987.

Schmidt-Schauß, M.: *Combination of Unification Algorithms in Arbitrary Disjoint Equational Theories.* SEKI-Report (submitted to 9th Conf. on Automated Deduction), Universität Kaiserslautern, 1987. To appear in J. of Symb. Comp., Special Issue on Unification (ed. C. Kirchner), 1989.

Shoenfield, J.R.: *Mathematical Logic.* Addison-Wesley, 1967.

Siekmann, J.H.: *Unification and Matching Problems.* Ph.D. Thesis, University of Essex, 1978.

Siekmann, J.H.: *Unification Theory. A Survey.* To appear in J. of Symb. Comp., Special Issue on Unification (ed. C. Kirchner), 1989.

Siekmann, J.H. & Szabo, P.: *The Undecidability of the DA-unification Problem.* SEKI-Report SR-86-19, Universität Kaiserslautern, 1986.

Smolka, G., Nutt, W., Goguen, J.A. & Meseguer, J.: *Order-Sorted Equational Computation.* SEKI-Report SR-87-14, Universität Kaiserslautern, 1987.

Stickel, M.E.: *A Complete Unification Algorithm for Associative-Commutative Functions.* Proc. of 4th Int. Joint Conf. on Art. Intelligence, Tblisi, 1975, p. 71-82.

Stickel, M.E.: *Unification Algorithms for Artificial Intelligence.* Ph. D. Thesis, Carnegie-Mellon

University, 1976.
Stickel, M.E.: *A Unification Algorithm for Associative-Commutative Functions.* JACM, Vol 28, No 3, 1981, p. 423-434.
Stickel, M.E.: *A Case Study of Theorem Proving by the Knuth-Bendix Method Discovering that $X^3 = X$ implies Ring Commutativity.* Proc. of 7th Conf. on Automated Deduction, Springer, LNCS 170, 1984, p. 248-258.
Stickel, M.E.: *Automated Deduction by Theory Resolution.* J. of Automated Reasoning, Vol1, No 4, 1985, p. 333-357.
Stickel, M.E.: *A Comparison of the Variable-Abstraction and Constant-Abstraction Methods for Associative- Commutative Unification.* J. of Automated Reasoning, Vol 3, No 3, 1987, p. 285-289.
Szabo, P.: *Unifikationstheorie erster Ordnung.* (In German), Dissertation, Universität Karlsruhe, 1982.
Tarski, A.: *A Remark on Functionally Free Algebras.* Ann. Math. (2), 47, 1946, p 163-165.
Taylor, W.: *Equational Logic.* Houston Journal of Mathematics 5, 1979.
Tiden, E.: *Unification in Combinations of Equational Theories.* Ph. D. Thesis, Stockholm, 1986.
Tiden, E.: *Unification in Combinations of Collapse-Free Theories with Disjoint Sets of Function Symbols.* Proc. of 8th Conf. on Automated Deduction, Springer, LNCS 230, 1986, p.431-450.
Wos, L., Overbeek, R., Lusk, E. & Boyle, J.: *Automated Reasoning - Introduction and Applications.* Prentice Hall, 1984.
Yellick, K.: *Combining Unification Algorithms for Confined Regular Equational Theories.* Proc. of Conf. on Rewriting Techniques and Applications, Springer, LNCS 202, 1985, p. 365-380. See also: Technical Report, MIT, 1985.